AFRL-RI-RS-TR-2018-078

SCHEDULING MISSION-CRITICAL FLOWS IN CONGESTED AND CONTESTED AIRBORNE NETWORK ENVIRONMENTS

STATE UNIVERSITY OF NEW YORK AT BUFFALO

MARCH 2018

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2018-078 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
MICHAEL J. MEDLEY
Work Unit Manager

/ S /
JOHN D. MATYJAS
Technical Advisor, Computing
& Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
MAY 2018	FINAL TECHNICAL REPORT	FEB 2014 – SEP 2017

4. TITLE AND SUBTITLE

SCHEDULING MISSION-CRITICAL FLOWS IN CONGESTED AND CONTESTED AIRBORNE NETWORK ENVIRONMENTS

5a. CONTRACT NUMBER
FA8750-14-1-0073

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62702F

6. AUTHOR(S)

Nicholas Mastronarde

5d. PROJECT NUMBER
T2CD

5e. TASK NUMBER
UB

5f. WORK UNIT NUMBER
NM

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University at Buffalo
State University of New York (SUNY)
501 Capen Hall
Buffalo, NY 14260

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RITF
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2018-078

12. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The U.S. Air Force requires timely, reliable, and resilient communications in adversarial network environments. Within these dynamic environments, network nodes experience congested and contested spectrum with only limited and intermittent bandwidth available to support communications. Although there are many techniques that can be leveraged across the network protocol stack to improve communication reliability, resilience, and spectral efficiency, delayed and lost packets are inevitable in such environments. Thus, scheduling the right packets at the right time becomes paramount. The objectives of this project are two-fold. The first objective is to establish new priority- and deadline-aware scheduling solutions to ensure that the highest priority network traffic, defined in the context of the mission, is reliably delivered to its destination when it is needed. The second objective is to develop a framework to evaluate different airborne networking and communications protocols in the context of the overlaying command and control, intelligence, surveillance, and reconnaissance (C2ISR) applications. To meet these objectives, contributions have been made towards optimal priority- and deadline-driven scheduling, delay-sensitive medium access control based on carrier-sense multiple access with collision avoidance (CSMA/CA), and the development of a framework that facilitates simulation-based and experimental airborne networking research and enables us to evaluate the effect of new communications and networking protocols on the mission itself. These contributions have been validated through a combination of simulation and experimental results.

15. SUBJECT TERMS

priority scheduling, latency scheduling, deadline-driven scheduling, delay-sensitive scheduling, markov decision process, airborne network communications, coverage path planning

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
					MICHAEL J. MEDLEY
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (*Include area code*)
U	U	U	UU	109	

Contents

1 Summary **1**

2 Introduction **2**

3 Priority and Deadline Driven Scheduling **6**
3.1 Introduction . . . 6
3.2 Methods, Assumptions and Procedures . . . 8
3.2.1 System model . . . 8
3.2.2 Formulation as a Markov Decision Process (MDP) . . . 10
3.2.3 Challenges and Existing Heuristics . . . 11
3.2.4 Structural Properties of the Optimal Policy . . . 13
3.2.4.1 Post-decision state dynamic programming . . . 13
3.2.4.2 Structural properties with respect to the traffic classes . . . 13
3.2.4.3 Structural properties with respect to the dynamics . . . 17
3.3 Results and Discussion . . . 17
3.3.1 Simulation setup . . . 17
3.3.2 Comparison to EDF, PQ, and WFQ . . . 18
3.3.3 Structural properties of the optimal scheduling policy . . . 19
3.3.4 Discussion . . . 22

4 Delay-Sensitive CSMA/CA Scheduling **24**
4.1 Introduction . . . 24
4.2 Methods, Assumptions and Procedures . . . 25
4.2.1 Physical Layer Model . . . 26
4.2.2 Data Link Layer Model . . . 27
4.2.3 Summary of the System's Operation . . . 27
4.2.4 Problem Formulation . . . 28
4.2.4.1 Multi-User Problem Formulation . . . 28
4.2.4.2 Multi-User Problem Decomposition . . . 29
4.2.5 Learning the Optimal Policy . . . 30
4.2.5.1 The Post-Decision State Learning Algorithm . . . 31
4.2.5.2 Virtual Experience Learning . . . 32
4.2.6 Rate-adaptive CSMA/CA Protocol . . . 32
4.3 Results and Discussion . . . 33
4.3.1 Impact of Arrival Rates and Holding Cost Constraints . . . 35

4.3.2 Comparison to Conventional CSMA/CA . . . 35
4.3.3 Discussion . . . 36

5 UB-ANC Drone: A Flexible Airborne Networking and Communications Testbed 37
5.1 Introduction . . . 37
5.2 Related Work . . . 38
5.3 Methods, Assumptions and Procedures . . . 39
5.3.1 Hardware Components . . . 39
5.3.2 Software Components . . . 40
5.3.2.1 Agent Control Unit (ACU) . . . 43
5.3.2.2 Network Control Unit (NCU) . . . 44
5.3.2.3 MAVLink Control Unit (MCU) . . . 46
5.3.2.4 Logging Unit (LU) . . . 47
5.4 Results and Discussion . . . 48

6 UB-ANC Emulator: An Emulation Framework for Multi-Agent Drone Networks 49
6.1 Introduction . . . 49
6.2 Related Work . . . 50
6.3 Methods, Assumptions and Procedures . . . 51
6.3.1 Software Architecture . . . 51
6.3.1.1 UB-ANC Agent . . . 52
6.3.1.2 MAV Object . . . 52
6.3.1.3 Emulation Engine . . . 52
6.3.1.4 Software in the Loop (SITL) . . . 52
6.3.2 Network Simulator Integration . . . 53
6.3.2.1 API for Network Simulator Integration . . . 53
6.3.2.2 Ns-3 Integration . . . 55
6.4 Results and Discussion . . . 56
6.4.1 UB-ANC Emulator Evaluation . . . 56
6.4.1.1 Accuracy . . . 57
6.4.1.2 Scalability . . . 59
6.4.1.3 Extensibility . . . 60
6.4.1.4 Simulating Other Parameters . . . 60
6.4.2 Ns-3 Integration Evaluation . . . 60
6.4.2.1 Node-to-Node Connectivity . . . 63
6.4.2.2 Network Connectivity (End-to-End) . . . 63
6.4.3 Discussion . . . 65

7 UB-ANC Planner: Energy Efficient Coverage Path Planning with Multiple Drones 68
7.1 Introduction . . . 68
7.2 Related Work . . . 69
7.3 Methods, Assumptions and Procedures . . . 70

7.3.1 Energy Consumption of Drone Flight . . . 70
7.3.2 Energy Efficient Coverage Path Planning For Multiple Drones . . 74
7.3.2.1 Problem Modeling . . . 74
7.3.2.2 Problem Formulation . . . 75
7.4 Results and Discussion . . . 78
7.4.1 Simulation Results . . . 78
7.4.1.1 Algorithm Scalability . . . 79
7.4.1.2 Energy Efficiency and Algorithm Adaptivity . . . 79
7.4.1.3 Multi-Drone Path Planning . . . 81
7.4.2 Experimental Evaluation . . . 81
7.4.3 Discussion . . . 84

8 Conclusion **88**

9 List of Acronyms **89**

List of Figures

1 System model. 7

2 Virtual queue illustration with $N = 2$ priority classes and $M = 3$ deadline classes. Bold arrows indicate the order in which packets are scheduled. (a) EDF schedules packets with earliest deadline while ignoring their priorities. (b) PQ schedules packets with highest priority while ignoring their deadlines. (c) WFQ allocates data rate to classes proportional to their priorities, and schedules packets within each priority class starting with the earliest deadline first. (d) Proposed solution (with illustrative scheduling order) accounts for both deadlines and priorities. 12

3 Comparison of normalized priority weighted throughputs for different load intensities. (a) $\alpha^1 = 1$ and $\alpha^2 = 1/4$. (b) $\alpha^1 = 1$ and $\alpha^2 = 3/4$. 18

4 Discount factor values at which optimal packet scheduling switches from τ_{12} class to τ_{21} class for different traffic and rate states. Here, r is the rate constraint, $(\alpha^1, \alpha^2) = (1, \frac{1}{2})$, and γ_{sw} is the switch-over discount factor which determines the urgency between classes τ_{12} and τ_{21}. 20

5 α^2_{sw} values at which packet scheduling switches from τ_{12} class to τ_{21} class for different cases. Here $\gamma = 0.98$, $\alpha^1 = 1$, and "-" indicates that the scheduling action does not change with α^2. 21

6 Optimal MDP-based policy schedules earliest deadline class τ_{21} when η is low and the higher priority class τ_{12} when η is high. Here $(\alpha^1, \alpha^2) = (1, \frac{1}{3})$ and $\gamma = 0.98$. 22

7 Comparison between the proposed virtual experience learning algorithm and PDS learning using the proposed rate-adaptive CSMA/CA protocol when users have heterogeneous arrival rates. Users 1, 2, and 3 have arrival rates 1, 2, and 3 packets/slot, respectively. All users have the same holding cost constraint (9 packets). (a) Cumulative average holding cost vs. time. (b) Cumulative average energy cost vs. time. 34

8 Comparison between the proposed virtual experience learning algorithm and PDS learning using the proposed rate-adaptive CSMA/CA protocol when users have heterogeneous holding cost contraints. Users 1, 2, and 3 have holding cost constraints 6, 9, and 12, respectively. All users have the same arrival rate (2 packets/slot). (a) Cumulative average holding cost vs. time. (b) Cumulative average energy cost vs. time. 35

9 Comparison between the proposed CSMA/CA protocol and the conventional CSMA/CA protocol (i.e., the IEEE 802.11 DCF) using virtual experience learning. Each point corresponds to one user's average energy cost and average holding cost averaged over ten 10,000 time slot simulations. The lines are included for reference and should not be interpreted as tradeoff curves. 10 users are simulated with holding cost constraints ranging from 3.5 to 10.25 packets. All users have the same arrival rate (0.5 packets/slot). 36

10 A UB-ANC drone (SDR configuration). 41
11 High-level software architecture diagram. (a) UB-ANC's core software architecture with its interface to the network. (b) SDR architecture with its interface to the Network Control Unit. (c) Standard wireless network architecture with its interface to the Network Control Unit. 45

12 The UB-ANC Emulator's software architecture. 51
13 Block diagram illustrating how we have integrated ns-3 into the UB-ANC Emulator. 54
14 APM Planner visualization for UB-ANC Emulator. 56
15 Comparison between emulation and experimentation for a three-drone mission. (a) Events vs. time; (b) Longitude vs. time; (c) Altitude vs. time. . . . 57
16 Potential sources of time shift between experiments and simulations for one drone. 58
17 UB-ANC Emulator resource usage for different numbers of emulated MAVs. 60
18 UB-ANC Emulator with two MAVs communicating over USRP N210 software-defined radios. 61
19 Energy consumption comparison for one drone. 61
20 Speed comparison for one drone. 62
21 Pressure changes comparison for one drone. 62
22 Number of packets received for 1000 packets sent. 63
23 A network of 7 MAVs visualized using APM Planner. MAV1 is the source and MAV7 is the destination. 64
24 Number of data packets sent by MAV1 and received by MAV7 under different APP rates and routing protocols. 66
25 Number of data and routing/overhead packets sent by each MAV under different APP rates and routing protocols. 67

26 A UB-ANC Drone with a custom frame, waypoint-based Pixhawk flight controller, Raspberry Pi 2, custom power sensor module, and 10,000 mAh battery. 70
27 Energy consumed as measured on the UB-ANC Drone for various patterns of flight. In Fig 27c, we omit the 180^o data point for line fitting as our planning does not allow re-visiting a node. It is shown here to demonstrate model validity. 72
28 Average power draw and time for different missions. 73
29 A cell and its neighbors, and the exterior angle for three nodes on the path. . 76
30 Illustration of the Lin-Kernighan Heuristic (LKH). 78

31 Run-time (max: 1 hr) and performance of MEPP on rectangular grids of different sizes without obstacles. 80
32 Rectangular grid maps used to evaluate the algorithms. Grey cells represent obstacles. 80
33 Comparison of algorithms over areas in Fig. 32. 81
34 Visualization of the planned paths for different algorithms in area 4 of Fig. 32. The drone's tour starts and ends in the upper-right corner of the grid. 82
35 UB North Campus. Areas dense with buildings are assumed to obstacles (shaded with diagonal lines). 83
36 Average path planning computation time (per drone), and energy consumption (per drone) for a set of drones covering UB North campus. Comparison between LKH and LKH-D algorithms. 83
37 Satellite view from UB stadium and the planned coverage paths. Virtual obstacles are shaded with diagonal lines. (a) Path planned by DLS algorithm. (b) Actual flight path in experiment. 85
38 Satellite view from UB stadium and the planned coverage paths. Virtual obstacles are shaded with diagonal lines. (a) Path planned by LKH-D algorithm. (b) Actual flight path in experiment. 86

List of Tables

1 List of notation. 9

2 Post-decision state learning algorithm at user i. 32
3 Simulation parameters. 34

4 Comparison between two UB-ANC drone configurations. 40
5 Abbreviated front-end APIs for the Network and MAVLink Control Units (i.e., the NCU and MCU). 42
6 An abbreviated list of MAVLink commands. 46
7 Parameters for the loiter command. 47
8 Abbreviated mission log. 48

9 API for integrating existing network simulation software into the UB-ANC Emulator. 55
10 Variance of experimental and simulation measurements across 5 rounds. . . . 59
11 Mean squared error (MSE) between the average experimental measurements and simulation measurements with (measured) and without (shifted) the extra arm-to-takeoff delay that appears in the simulations. 59
12 Transmitted and received data rates (bytes/s) at MAV1 and MAV7, respectively, excluding overheads. 65

13 Simulation statistics for area 4 in Fig. 32. 82
14 Experimental statistics for Area 4 in Fig. 32. 84

1. *Summary*

The U.S. Air Force requires timely, reliable, and resilient communications in adversarial network environments. Within these dynamic environments, network nodes experience congested and contested spectrum with only limited and intermittent bandwidth available to support communications. Although there are many techniques that can be leveraged across the network protocol stack to improve communication reliability, resilience, and spectral efficiency, delayed and lost packets are inevitable in such environments. Thus, scheduling the right packets at the right time becomes paramount. The objectives of this project are two-fold. The first objective is to establish new priority- and deadline-aware scheduling solutions to ensure that the highest priority network traffic, defined in the context of the mission, is reliably delivered to its destination when it is needed. The second objective is to develop a framework to evaluate different airborne networking and communications protocols in the context of the overlaying command and control, intelligence, surveillance, and reconnaissance (C2ISR) applications. To meet these objectives, over the duration of the project (43 months), contributions have been made towards optimal priority- and deadline-driven scheduling, delay-sensitive medium access control based on carrier-sense multiple access with collision avoidance (CSMA/CA), and the development of a framework that facilitates simulation-based and experimental airborne networking research and enables us to evaluate the effect of new communications and networking protocols on the mission itself. These contributions have been validated through a combination of simulation and experimental results.

Approved for Public Release; Distribution Unlimited.

2. *Introduction*

Mission-critical airborne networking and communications (ANC) applications, such as command and control, intelligence, surveillance, and reconnaissance (C2ISR) are characterized by multiple time-varying traffic flows (C2 signaling, video, audio, etc.) comprising heterogeneous packets with different deadlines and priorities. Moreover, these applications must operate in congested and contested spectral environments with only limited/intermittent bandwidth to support communications. This mismatch between the limited network resources and the stringent requirements of the overlaying applications poses a significant challenge in the design of optimal single- and multi-user scheduling algorithms.

In parallel, the complexity of ANC applications is expected to increase by orders of magnitude as C2ISR operations expand from dozens of network nodes [1] (AWACS, JSTARS, etc.) to include swarms of 100s or 1000s of low cost attritable small unmanned aircraft systems [2,3] (UAS[1]). In this context, many techniques can be leveraged across the network protocol stack to improve throughput, latency, reliability, and spectral efficiency; however, these metrics alone cannot tell us how well the underlying ANC capabilities support the overlaying C2ISR applications. In other words, given two or more possible protocol stacks, these metrics cannot tell us which one will work best for a specific mission in a specific network environment. This is because multi-agent ANC systems operate not only in the "cyber" domain, where these metrics are meaningful, but also in the "physical" domain, where performance must be measured based on the physical behavior of the agents within the system (whether or not the airborne nodes accomplish their mission, if it is accomplished in a timely manner, etc.). Once we see that multi-agent ANC systems are in fact *cyber-physical systems* [4], then we can no longer look at the underlying networking and communications capabilities (i.e., the cyber component) in isolation.

The objectives of this project are two-fold and address relevant challenges for the U.S. Air Force. The first objective is to establish new priority- and deadline-aware scheduling solutions to ensure that the highest priority network traffic, defined in the context of the mission, is reliably delivered to its destination when it is needed. The second objective is to develop a framework to evaluate different airborne networking and communications protocols in the context of the overlaying C2ISR applications. To meet these objectives, over the duration of the project (43 months), contributions have been made towards optimal priority- and deadline-driven scheduling, delay-sensitive medium access control based on carrier-sense multiple access with collision avoidance (CSMA/CA), and the development of a framework that facilitates simulation-based and experimental airborne networking research

[1] In this report, we will use the terms UAS, unmanned aerial vehicle (UAV), micro aerial vehicle (MAV), and drone interchangeably.

Approved for Public Release; Distribution Unlimited.

and enables us to evaluate the effect of new communications and networking protocols on the mission itself. In particular, we have made the following contributions.

- **Priority and deadline driven scheduling (Section 3):** We formulate the point-to-point scheduling at a congested network node as a Markov decision process (MDP) that considers the deadlines and priorities of each packet as well as the dynamic packet arrivals and channel conditions. Within this framework, we formulate the problem with the objective of maximizing the node's long-run priority-weighted throughput (i.e., the sum of priorities of all packets that are successfully transmitted) subject to instantaneous transmission rate constraints. We then analyze the structural properties of the optimal scheduling policy with respect to the deadlines and priorities of the backlogged packets. Additionally, we compare our approach to existing heuristics such as Priority Queuing (PQ), Earliest Deadline First (EDF), and Weighted Fair Queuing (WFQ). Lastly, we experimentally show that the optimal scheduling policy has a switch-over type structure in several key parameters including the relative priorities of different traffic classes, the discount factor, and the traffic load intensity. This work has been published in [5].

- **Delay-Sensitive CSMA/CA Scheduling (Section 4):** In Section 3, we considered single-user point-to-point scheduling. In this section, we investigate energy-efficient scheduling of delay-sensitive data over fading channels. To tradeoff energy and delay, we combine adaptive rate transmission at the physical layer with a rate-adaptive medium access control (MAC) protocol based on carrier sense multiple access with collision avoidance (CSMA/CA). We formulate the multi-user scheduling problem as a constrained Markov decision process (CMDP). We show that the multi-user problem is intractable and propose to decompose it into multiple (coupled) single-user problems. We design a reinforcement learning algorithm to solve the single-user problems online so that users can achieve energy-efficient operation while meeting their delay constraints, even though the channel, traffic, and multi-user dynamics are unknown a priori. Our proposed MAC protocol enables users to meet significantly tighter delay constraints while also consuming less energy than under the 802.11 Distributed Coordination Function (DCF). Moreover, the proposed learning algorithm converges significantly faster than a state-of-the-art solution. This work has been published in [6] and an extended version of the work is being prepared for journal publication.

- **UB-ANC Drone: A flexible airborne networking and communications framework (Section 5):** In this section, we introduce the *UB-ANC Drone* and the *UB-ANC Agent* software. UB-ANC Drone is an open software/hardware platform that aims to facilitate rapid testing and repeatable comparative evaluation of airborne networking and communications protocols at different layers of the protocol stack. It combines quadcopters capable of autonomous flight with sophisticated command and control capabilities and embedded software-defined radios (SDRs), which enable flexible deployment of novel communications and networking protocols. This is in contrast to existing airborne network testbeds, which only support standard inflexible wireless technologies, e.g., Wi-Fi or Zigbee. UB-ANC Drone is designed with emphasis on modularity and extensibility, and is built around popular open-source projects and

Approved for Public Release; Distribution Unlimited.

standards developed by the research and hobby communities. This makes it highly customizable, while also simplifying its adoption.

The **UB-ANC Agent**[2], which is the software that controls the drone, is designed to be compatible with any flight controller that supports the popular Micro Air Vehicle Communications Protocol (MAVLink[3]). With its modular design, UB-ANC Drone provides tools for networking researchers to study airborne networking protocols and robotics researchers to study mission planning algorithms without worrying about other implementation details. Furthermore, we envision that it will facilitate collaborative work between networking and robotics researchers interested in problems related to network topology control and managing trade offs between mission objectives and network performance. This work has been published in [7].

- **UB-ANC Emulator: An emulation framework for multi-agent drone networks (Section 6):** The *UB-ANC Emulator* is an emulation environment created to design, implement, and test various applications (missions) involving one or more drones in software, and provide seamless transition to experimentation. UB-ANC Emulator provides flexibility in terms of the underlying flight dynamics and network simulation models. By default, it provides low-fidelity flight dynamics and network simulation, thus high scalability (it can support a large number of emulated agents). Depending on the application, it can connect to a high-fidelity physics engine for more accurate flight dynamics of agents (drones). It can also connect to a high-fidelity network simulation to model the effect of interference, packet losses, and protocols on network throughput, latency, and reliability. For example, we have integrated ns-3 into the emulator. Another important aspect of the UB-ANC Emulator is its ability to be extended to different setups and connect to external communication hardware. This capability allows robotics researchers to emulate the mission planning part in software while the network researcher tests new network protocols on real hardware, or allows a network of real drones to connect to emulated drones and coordinate their tasks. This work has been published in [8–10] and an extended version of the work is being prepared for journal publication.

- **UB-ANC Planner: Energy-efficient coverage path planning with multiple drones (Section 7):** Utilizing the UB-ANC Drone and UB-ANC Emulator, we developed the *UB-ANC Planner* to demonstrate the framework's sophisticated mission planning capabilities. To this end, we consider the problem of covering an arbitrary area containing obstacles using multiple drones, i.e., the so-called *coverage path planning* (CPP) problem. The goal of the CPP problem is to find paths for each drone such that the entire area is covered. However, a major limitation in such deployments is drone flight time. To most efficiently use a swarm, we propose to minimize the maximum energy consumption among all drones' flight paths. We perform measurements to understand energy consumption of a drone. Using these measurements, we formulate

[2]https://github.com/jmodares/UB-ANC-Agent

[3]MAVLink is a protocol that is used to package command and control messages directed to the flight controller and package telemetry information sent from the flight controller.

Approved for Public Release; Distribution Unlimited.

an Energy Efficient Coverage Path Planning (EECPP) problem. We solve this problem in two steps: a load-balanced allocation of the given area to individual drones, and a minimum energy path planning (MEPP) problem for each drone. Results show that our algorithm is more computationally efficient and provides more energy-efficient solutions compared to the other heuristics. This work has been published in [11] and an extended version of the work is being prepared for journal publication.

In the remainder of this report, we present the Methods, Assumptions and Procedures, and Results and Discussion for each sections described above. We conclude the report in Section 8.

Approved for Public Release; Distribution Unlimited.

3. *Priority and Deadline Driven Scheduling*

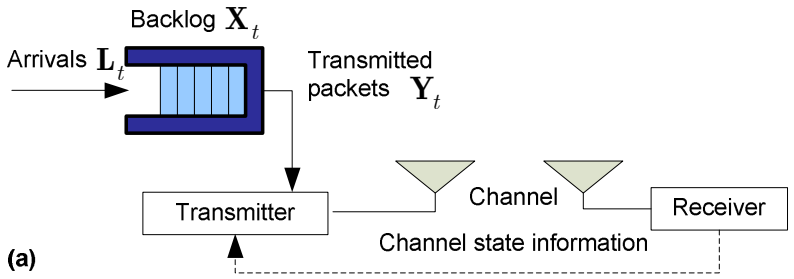
3.1 Introduction

With the evolution of wireless technology, civilian applications, such as multimedia streaming, video conferencing, remote monitoring, and online gaming, have gained widespread interest and military applications, such as C2ISR, have become increasingly important. These applications comprise possibly multiple information flows (e.g., C2 signals, video, and audio), which contain packets with different priorities and deadlines, and result in time-varying traffic loads. At the same time, data for such applications is transmitted in dynamic wireless environments with limited physical resources where they experience time-varying and a priori unknown channel conditions. This poses a challenge in designing optimal scheduling solutions. In this section, we consider the problem of optimal point-to-point scheduling of traffic with different deadlines and priorities. We consider a fairly general framework where this traffic could be generated locally by the node itself or could be a combination of its own traffic and traffic received from other nodes in the network.

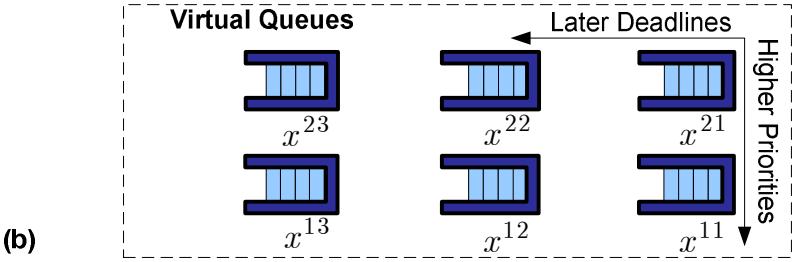
There is a vast literature on scheduling in wireless networks, e.g., [12–29]; however, existing solutions often ignore one or more of the unique requirements of multimedia applications. For example, in [13, 16], fluid-based models ignore the deadlines of the packets; and, while delay constraints of individual packets or groups of packets are considered in [15, 17], neither [15, 17] nor [13, 16] consider packets with different priorities. However, scheduling packets with different priorities has been studied [12, 14, 26–28]. For example, to take into account traffic with heterogeneous priorities and deadlines, a rate-distortion optimization framework called RaDiO was introduced in [14]. However, the RaDiO framework does not provide intuition about the structure of the scheduling policy. In [29], a scheduling policy called CD^2, which considers channel conditions, deadlines, and distortion, is developed for a multiuser downlink scenario. Under the assumptions that a new frame arrives only after the previous one is transmitted, that only one packet per time slot is scheduled, and that channel conditions remain constant over the optimization horizon, [29] finds that the optimal scheduling policy for two users is of switch-over type, and that the optimal multi-user policy can be determined through pair-wise comparisons. Unlike [29], we do not make such assumptions on packet arrivals, transmissions, or channel conditions, and we consider point-to-point scheduling instead of multi-user downlink scheduling.

The most closely related work to ours is [28], which considers many of the same requirements as we do, but focuses on the transmission of a single video sequence with a periodic traffic structure. While we take inspiration from the models and theory outlined in that paper, there are several key differences between our work and [28]. First, we consider

Approved for Public Release; Distribution Unlimited.



(a) Transmission model.



(b) Classification of backlogged packets into virtual queues.

Figure 1: System model.

scheduling (possibly multiple) non-periodic flows over a wireless node. Second, while [28] considers energy-constrained transmission with adaptive power control, we consider rate-constrained transmission, where the rates are determined by the lower layers (e.g., data link and physical layers). This is important because it implies that our solution can be applied regardless of the algorithms implemented at the lower layers to determine the transmission rates. For example, our solution could be integrated into a sophisticated cognitive radio system such as ROSA [30]. Third, while our main theoretical result on the structure of the optimal scheduling policy is similar to [28], we provide a more detailed and complete proof, and also make appropriate changes due to the different constraints in our problem. Finally, we experimentally investigate how the structure of the optimal policy depends on parameters other than the absolute deadlines and priorities of different traffic classes.

Many existing solutions react to the experienced network dynamics in a "myopic" way, by optimizing the transmission strategies based only on information about the current traffic and channel condition [12, 13, 20, 26]. However, our prior work in [21, 25, 27, 31] shows that significant improvements in resource utilization and performance can be achieved using "foresighted" scheduling strategies that account for the fact that current decisions impact both the immediate and future network performance.

In summary, existing solutions either (i) rely on simplistic traffic models that ignore the different deadlines and priorities of packets or (ii) perform myopic optimizations that ignore the impact of current scheduling decisions on the future performance.

Our contributions are as follows:

1. We formulate the scheduling problem as a Markov decision process (MDP) that takes into account the delay deadlines and priorities of individual packets, the random traffic

Approved for Public Release; Distribution Unlimited.

loads, and the dynamic channel conditions. The objective of the MDP is to maximize the congested node's long-run priority-weighted throughput subject to instantaneous transmission rate constraints.

2. We analyze the structural properties of the optimal scheduling policy. We show theoretically that it is possible to determine the order with which to schedule some packets based only on their absolute deadlines and priorities.

3. We study experimentally how the optimal scheduling policy depends on different parameters including the relative priorities of different packets, the discount factor, and the traffic load intensity.

4. Since we use rate constrained optimization, our scheduling problem formulation is compatible with any approach used at the lower layers to determine the transmission rate. Moreover, our theoretical and experimental observations translate to any communication system, regardless of how the transmission rates are determined.

We note that we cannot solve the general scheduling problem using our MDP formulation because of the curse of dimensionality (that is, the problem's complexity increases exponentially in the number of considered deadline and priority classes). Instead, in this project, we aim to reveal some properties of the optimal scheduling policy, which we believe can be exploited to find new low-complexity scheduling heuristics that outperform existing heuristics such as Priority Queueing (PQ), Earliest Deadline First (EDF), and Weighted Fair Queueing (WFQ).

The remainder of this section is organized as follows. In Sections 3.2.1 and 3.2.2, we model and formulate the scheduling problem, respectively. In Section 3.2.3, we discuss the problem's challenges and discuss existing heuristics. In Section 3.2.4, we analyze the structural properties of the scheduling policy. In Section 3.3, we present our experimental results. We conclude in Section 3.3.4.

3.2 Methods, Assumptions and Procedures

3.2.1 System model

We consider a point-to-point wireless communication system as illustrated in Fig. 1a. We assume that time is slotted into discrete-time intervals of length Δt, which is a short interval of time over which different numbers of packets are transmitted depending on the channel conditions. We propose to classify the backlogged packets into N priority classes and M deadline classes as illustrated in Fig. 1b. We assume that $i = 1$ (resp. $i = N$) corresponds to the highest (resp. lowest) priority class and that packets in deadline class j must be delivered to their destination within $d^j = j\Delta t$ seconds. We define traffic class τ_{ij} to contain all packets in priority class $i \in \{1, \ldots, N\}$ and deadline class $j \in \{1, \ldots, M\}$. A list of notation is provided in Table 1.

We denote the backlog of traffic class τ_{ij}'s virtual queue in slot t as $x_t^{ij} \in \{0, 1, \ldots\}$ and assume that $l_t^{ij} \in \{0, 1, \ldots\}$ new packets arrive in τ_{ij}'s virtual queue at the end of slot t.

Approved for Public Release; Distribution Unlimited.

Table 1: List of notation.

Δt, t	Length of the time slot, time slot index
N, M	Number of priority classes, number of deadline classes
i, j, τ_{ij}	Priority class i, deadline class j, traffic class ij
α^i, d^j	Priority of class i, deadline of class j
x^{ij}, l^{ij}, y^{ij}, r	Number of backlogged packets, packet arrivals, packets transmitted, rate
X, L, Y	Traffic state matrix, packet arrival matrix, scheduling action matrix
$\tilde{X}$, $\tilde{V}(X,r)$	Post-decision state, post-decision state value function
γ	Discount factor
Q	Lower shift matrix
$p_X(X_{t+1}\|X_t,Y_t)$, $p_r(r_{t+1}\|r_t)$, $p\,(s_{t+1}\|s_t,Y_t)$	Traffic state, rate state, and overall transition probability
$p_{l^{ij}}(l_t^{ij})$	Packet arrival distribution
$u_{ij}(x_t^{ij},y_t^{ij})$, $u(X_t,Y_t)$	Utility of a class, composite utility
$V(s)$, $Q(s,Y)$, $\pi(s)$	Opt. state value function, opt. action-value function, opt. policy
η, Ψ	Load intensity, normalized priority-weighted throughput

We assume that class τ_{ij} packet arrivals are independent and identically distributed with distribution $p_{l^{ij}}(l_t^{ij})$. The buffer state x_t^{ij} evolves recursively as follows:

$$x_{t+1}^{ij} = x_t^{i,j+1} - y_t^{i,j+1} + l_t^{ij}, \text{ for all } \tau_{ij} \tag{3.1}$$

where $0 \leq y_t^{ij} \leq x_t^{ij}$ is the number of packets transmitted from τ_{ij}'s virtual queue in slot t and $x_t^{i,M+1} = y_t^{i,M+1} = 0$. All packets in priority class i and deadline class $j+1$ that are not transmitted in slot t transition to deadline class j in slot $t+1$. If packets in deadline class 1 are not transmitted, then they expire due to deadline violation. Note that, for ease of exposition, (3.1) assumes lossless transmission; however, lossy transmission can be included in the model as in, e.g., [25].

We define the composite virtual queue state as a matrix X_t with elements x_t^{ij} (hereafter, referred to as the traffic state) and scheduling decision as a matrix Y_t with elements y_t^{ij}, and the packet arrivals as a matrix L_t with elements l_t^{ij}. Each matrix is of size $N \times M$. The buffer recursion defined in (3.1) can be rewritten in matrix form as follows:

$$X_{t+1} = (X_t - Y_t)Q + L_t, \tag{3.2}$$

where Q is an $M \times M$ lower shift matrix with ones on the sub-diagonal and zeros elsewhere.

Based on the buffer recursion in (3.2), the sequence of traffic states $\{X_t : t = 0, 1, \ldots\}$ can be modeled as a controlled Markov chain with transition probabilities $p_X(X_{t+1}|X_t,Y_t)$, where

$$p_X(X_{t+1}|X_t,Y_t) = \sum_{L\in\{0,1,\ldots\}^{N\times M}} \prod_{i=1}^{N}\prod_{j=1}^{M} \left\{ \begin{array}{l} I_{\{x_{t+1}^{ij} = x_t^{i,j+1} - y_t^{i,j+1} + l_t^{ij}\}} \\ \times p_{l^{ij}}(l_t^{ij}) \end{array} \right\} \tag{3.3}$$

Approved for Public Release; Distribution Unlimited.

and $I_{\{.\}}$ is an indicator function.

When y_t^{ij} packets are transmitted from traffic class τ_{ij} in slot t, the immediate utility received by the transmitter is denoted by $u_{ij}(x_t^{ij}, y_t^{ij}) \geq 0$. We assume that the composite utility of all virtual queues has an additive form similar to, e.g., [14] and [29]:

$$u(X_t, Y_t) = \sum_{i=1}^{N} \sum_{j=1}^{M} u_{ij}(x_t^{ij}, y_t^{ij}). \tag{3.4}$$

For example, in this project, we optimize a priority-weighted throughput. This is equivalent to defining the utility as $u_{ij}(x_t^{ij}, y_t^{ij}) = \alpha^i y_t^{ij}$, for all τ_{ij}, where α^i is the relative priority of class i packets with $\alpha^i > \alpha^{i+1} > 0$.

We assume that the lower layers determine the transmission rate constraint seen by the application layer. This is in contrast to [28] which jointly optimizes scheduling and power control. We believe our model is more widely applicable because it does not make any assumptions about the lower layers of the protocol stack. For example, it could be easily applied to packet scheduling over a cognitive radio link where the transmission rate is determined as in, e.g., [30].

Let $r_t \in \mathcal{R}$ denote the transmission rate (in packets/time slot) that can be achieved over the point-to-point link in slot t. We assume that $\mathcal{R}$ is discrete and finite and that the sequence $\{r_t : t = 0, 1, \ldots\}$ can be modeled as a Markov chain with transition probability function $p_r(r_{t+1}|r_t)$, which reflects the fact that the channel conditions are correlated from one time slot to the next. The choice of scheduling action Y_t is constrained by r_t, i.e., $\sum_{i=1}^{N} \sum_{j=1}^{M} y_t^{ij} \leq r_t$. We assume that r_t is known at the beginning of each time slot.

3.2.2 Formulation as a Markov Decision Process (MDP)

In this section, we introduce the scheduling problem formulation. We define the state of the system at time t as $s_t = (X_t, r_t)$. The sequence of states $\{s_t : t = 0, 1, \ldots\}$ can be modeled as a controlled Markov chain with transition probability function

$$p\left(s_{t+1}|s_t, Y_t\right) = p_X(X_{t+1}|X_t, Y_t) p_r(r_{t+1}|r_t) \tag{3.5}$$

The objective of the scheduling problem is to determine the transmission action in each slot to maximize the long-run utility subject to a transmission rate constraint in each slot: i.e.,

$$\max_{Y_t, \forall t} E\left[\sum_{t=0}^{\infty} \gamma^t u(X_t, Y_t)\right] \text{ s.t. } 0 \leq Y_t \leq X_t \text{ and } \sum_{i=1}^{N} \sum_{j=1}^{M} y_t^{ij} \leq r_t, \tag{3.6}$$

where $\gamma \in [0, 1)$ is a discount factor, $Y_t \leq X_t$ denotes element-wise inequality, and the expectation is taken over the sequence of states. The discount factor determines the relative importance of the immediate and the expected future utilities.

The optimal solution to (3.6) satisfies the following Bellman equation:

$$V(s) = \max_{\substack{0 \leq Y \leq X, \\ \sum_{i=1}^{N} \sum_{j=1}^{M} y^{ij} \leq r}} \underbrace{\left\{ \begin{array}{l} u(X, Y) + \\ \gamma \sum_{X', r'} \left[\begin{array}{l} p([X', r']|[X, r], Y) \\ V([X', r']) \end{array} \right] \end{array} \right\}}_{Q(s,Y)}, \forall s = (X,r). \tag{3.7}$$

Approved for Public Release; Distribution Unlimited.

We refer to $V(s)$ as the optimal state-value function and $Q(s,Y)$ as the optimal action-value function. If the utility and transition probability functions are known, then V can be determined using the well-known value iteration or policy iteration algorithms [32]. Subsequently, the optimal policy $\pi(s)$, which gives the optimal action to take in each state, can be determined as:

$$\pi(s) = \max_{0 \leq Y \leq X,\ \sum_{i=1}^{N} \sum_{j=1}^{M} y^{ij} \leq r} Q(s, Y). \tag{3.8}$$

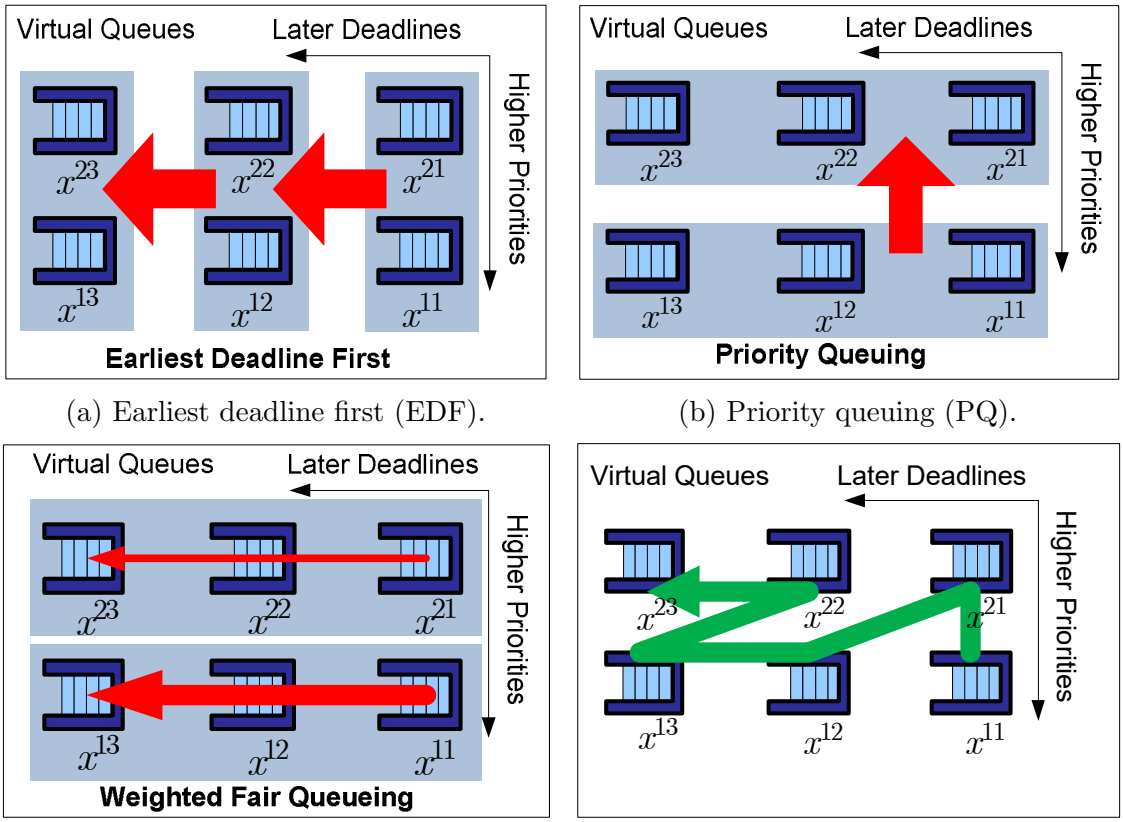
In [27] we have shown that, when optimizing a priority-weighted throughput metric as in (3.6), it is optimal to myopically optimize the channel rates (i.e., always maximize instantaneous channel rate); therefore, the channel rates do not have to be determined within the MDP. This implies that we can apply the proposed scheduling approach with any cross-layer strategies for determining the transmission rates.

3.2.3 Challenges and Existing Heuristics

We cannot solve the general scheduling problem using value iteration because it suffers from the curse of dimensionality, i.e., its complexity increases exponentially in $N \times M$ (e.g., for $N = 10$ and $M = 100$, and two buffer states per traffic class, there are 2^{1000} possible states). For this reason, we use the MDP formulation as a springboard for analyzing the structure of the optimal policy and to work towards developing lower complexity heuristics that exploit the structure of the optimal solution. While we can show that packets in class τ_{ij} should be transmitted (i) before packets in classes with the same deadline, but lower priority, (ii) before packets in classes with the same priority, but later deadlines, and (iii) before packets with lower priorities and later deadlines (see Lemma 2 in Section 3.2.4.2 for all of these cases), it is non-obvious which order we should transmit packets with earlier deadlines *and* lower priorities. This is because the optimal transmission order of these packets depends on the packet arrival distributions of the various traffic classes, their relative priorities, and the channel dynamics.

Suboptimal scheduling heuristics exist that simplify the scheduling decision. For example, the EDF heuristic ignores priority, and [see Fig. 2a] performs suboptimally except in uncongested networks where there is time to transmit all packets [31]. In contrast, the PQ heuristic ignores deadlines [see Fig. 2b], and performs suboptimally except in highly congested networks where there is only time to transmit the highest priority packets [31]. Weighted Fair Queueing (WFQ) allocates data rate to different priority classes (flows) proportional to their priorities (weights) [see Fig. 2c]. Although WFQ schedules packets from earliest deadline to latest deadline within a priority class, it is suboptimal because it ignores the packet arrival distribution and the traffic load intensity when making scheduling decisions. For example, in a congested network, WFQ allocates too much rate to lower priority packets. Since our proposed solution adapts the scheduling order based on the packet priorities and packet deadlines, and arrival and channel dynamics, it achieves optimal performance in uncongested and congested environments [see Fig. 2d]. Furthermore, if there are more traffic classes, then the optimal scheduling order based on our MDP approach will be more complex and differ significantly from the scheduling orders resulting from EDF, PQ, and WFQ; therefore, we expect that these heuristics will perform increasingly worse compared to the optimal solution as the number of traffic classes increases.

Approved for Public Release; Distribution Unlimited.



(a) Earliest deadline first (EDF).

(b) Priority queuing (PQ).

(c) Weighted fair queuing (WFQ).

(d) Proposed.

Figure 2: Virtual queue illustration with $N = 2$ priority classes and $M = 3$ deadline classes. Bold arrows indicate the order in which packets are scheduled. (a) EDF schedules packets with earliest deadline while ignoring their priorities. (b) PQ schedules packets with highest priority while ignoring their deadlines. (c) WFQ allocates data rate to classes proportional to their priorities, and schedules packets within each priority class starting with the earliest deadline first. (d) Proposed solution (with illustrative scheduling order) accounts for both deadlines and priorities.

Approved for Public Release; Distribution Unlimited.

3.2.4 Structural Properties of the Optimal Policy

In this section, we analyze the structure of the optimal scheduling policy. We introduce the concept of a post-decision state in Section 3.2.4.1 and analyze the optimal scheduling policy's structure in Section 3.2.4.2.

3.2.4.1 Post-decision state dynamic programming

We define a post-decision state as an intermediate state that occurs after the scheduling action is applied, but before the new packet arrivals and rate transition occur. The post-decision state separates the known information about the state transition from the unknown information [25]. In particular, the post-decision rate state in time slot t, denoted by $\tilde{r}_t$, is the same as the conventional rate state at time t, i.e., $\tilde{r}_t = r_t$. This is because the rate state transition is statistically independent of the scheduling action. Meanwhile, the post-decision state of class τ_{ij}'s virtual queue, denoted by $\tilde{x}_t^{ij}$, is defined as follows:

$$\tilde{x}_t^{ij} = x_t^{i,j+1} - y_t^{i,j+1}. \tag{3.9}$$

The next state of class τ_{ij}'s virtual queue can be determined from its post-decision state as $x_{t+1}^{ij} = \tilde{x}_t^{ij} + l_t^{ij}$. Rewriting (3.9) in matrix form, we obtain:

$$\tilde{X}_t = (X_t - Y_t)Q \tag{3.10}$$

where $\tilde{X}_t$ is the post-decision traffic state, which is an $N \times M$ matrix with elements $\tilde{x}_t^{ij}$. It follows that the next traffic state can be determined from the post-decision traffic state as $X_{t+1} = \tilde{X}_t + L_t$. We can define a post-decision state value function, denoted by $\tilde{V}(X, r)$, using the conventional value function, and vice versa:

$$\tilde{V}(X,r) = \sum_{r' \in \mathcal{R}} p_r(r'|r) \sum_{L \in \{0,1,\dots\}^{N \times M}} \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\begin{array}{l} p_{l^{ij}}(l^{ij}) \\ \times V(X + L, r') \end{array} \right] \tag{3.11}$$

$$V(X,r) = \max_{0 \le Y \le X,\ \sum_{i=1}^{N} \sum_{j=1}^{M} y^{ij} \le r} u(X,Y) + \gamma \tilde{V}\left((X-Y)Q, r\right) \tag{3.12}$$

For brevity, we will write (3.11) as $\tilde{V}(X, r) = E[V(X + L, r')]$, where the expectation is over the rate transition and arrival distributions.

3.2.4.2 Structural properties with respect to the traffic classes

In this section, we investigate the structural properties of the optimal policy. Recall that packets in class τ_{ij} have deadline d^j, where $d^M > \cdots > d^1 > 0$, and priority α^i, where $\alpha^1 > \cdots > \alpha^N > 0$. We introduce the following definition.

Definition 1 (Scheduling urgency)**.** *In any time slot t, if $(x_t^{ij} - y_t^{ij,*})y_t^{k\ell,*} = 0$ for all $x_t^{ij} > 0$ and for all rate states r_t such that $\sum_{a=1}^{N} \sum_{b=1}^{M} x_t^{ab} > r_t$, then class τ_{ij} has a higher scheduling urgency than class $\tau_{k\ell}$. We denote this relationship by $\tau_{ij} \triangleleft \tau_{k\ell}$.*

Approved for Public Release; Distribution Unlimited.

Note that, if $\sum_{a=1}^{N}\sum_{b=1}^{M} x_t^{ab} \leq r_t$, then all the packets can be scheduled and the scheduling urgency between classes is not important anymore. The scheduling urgency is analogous to the concept of transmission priority defined in [28], but has a slightly different definition because of the rate constraint.

The following lemma shows that the classes can be prioritized based on the optimal post-decision state value function. This result is analogous to Lemma 1 in [28].

Lemma 1. *For any two classes τ_{ij} and $\tau_{k\ell}$, if*

$$\alpha^i + \gamma\tilde{V}\left((X + A^{k\ell})Q, \tilde{r}\right) > \alpha^k + \gamma\tilde{V}\left((X + A^{ij})Q, \tilde{r}\right), \forall X \tag{3.13}$$

where A^{ij} is an $N \times M$ matrix with element $a^{ij} = 1$ and all other elements equal to 0, and $\sum_{a=1}^{N}\sum_{b=1}^{M} x^{ab} > r$, then $\tau_{ij} \lhd \tau_{k\ell}$.

Proof. We prove the lemma by contradiction. Suppose that (3.13) holds. Additionally, assume that the optimal scheduling action in state (X, r) is denoted by Y^* and that $\tau_{ij} \not\lhd \tau_{k\ell}$. Since $\sum_{a=1}^{N}\sum_{b=1}^{M} x^{ab} > r$, the optimal scheduling action Y^* will exactly satisfy the rate constraint, i.e., $\sum_{a=1}^{N}\sum_{b=1}^{M} y^{ab,*} = r$. Taking these facts together, it follows that $(x^{ij} - y^{ij,*})y^{k\ell,*} \neq 0$ and therefore $(x^{ij} - y^{ij,*}) > 0$ and $y^{k\ell,*} > 0$. Consider another scheduling action $Y = Y^* + A^{ij} - A^{k\ell}$. It is clear that the value of the optimal action Y^* is $V(X, r)$. Subtracting $V(X, r)$ from the value of Y we obtain:

$$\begin{aligned}
&u(X, Y) + \gamma\tilde{V}\left((X - Y)Q, \tilde{r}\right) - V(X, r) \\
&= u(X, Y) + \gamma\tilde{V}\left((X - Y)Q, \tilde{r}\right) - \left[u(X, Y^*) + \gamma\tilde{V}\left((X - Y^*)Q, \tilde{r}\right)\right] \\
&= \alpha^i - \alpha^k + \gamma\tilde{V}\left((X - Y^* - A^{ij} + A^{k\ell})Q, \tilde{r}\right) - \gamma\tilde{V}\left((X - Y^*)Q, \tilde{r}\right),
\end{aligned} \tag{3.14}$$

where the second equality follows from the fact that $u(X, Y) - u(X, Y^*) = \alpha^i - \alpha^k$, and $Y = Y^* + A^{ij} - A^{k\ell}$. We may rewrite the final line in (3.14) as follows:

$$\alpha^i - \alpha^k + \gamma\tilde{V}\left((X' + A^{k\ell})Q, \tilde{r}\right) - \gamma\tilde{V}\left((X' + A^{ij})Q, \tilde{r}\right) \tag{3.15}$$

where $X' = X - Y^* - A^{ij}$. It follows from (3.13) that (3.15) is greater than 0, which contradicts our assumption that Y^* is the optimal scheduling action. □ □

Unfortunately, the post-decision state value function may not be known because it is too complex to compute, or because the traffic arrival and channel dynamics are unknown; therefore, it is not always possible to use Lemma 1 to determine the optimal scheduling urgency. We show in Lemma 2 that we can determine the scheduling urgency for some (but not all) traffic classes based only on their deadlines and priorities.

Lemma 2. *If $\alpha^i \geq \alpha^k$ and $d^j \leq d^\ell$ (equalities do not hold at the same time), then $\tau_{ij} \lhd \tau_{k\ell}$.*

Proof. To simplify the notation in the bulk of the proof, we will prove that $\tau_{i,j+1} \lhd \tau_{k,\ell+1}$ for all $j \in \{0, \ldots, M-1\}$.

To prove that $\tau_{i,j+1} \lhd \tau_{k,\ell+1}$, we need to show that if $\alpha^i \geq \alpha^k$ and $d^{j+1} \leq d^{\ell+1}$ (equalities do not hold at the same time), then

$$\alpha^i + \gamma\tilde{V}\left((X + A^{k,\ell+1})Q, \tilde{r}\right) > \alpha^k + \gamma\tilde{V}\left((X + A^{i,j+1})Q, \tilde{r}\right), \ \forall X. \tag{3.16}$$

Approved for Public Release; Distribution Unlimited.

We first show that (3.16) holds for traffic classes τ_{i1} and $\tau_{k,\ell-j+1}$ for all $\ell \geq j$. We then show that it holds for all classes $\tau_{i,j+1}$ and $\tau_{k,\ell+1}$ for all $\ell \geq j$. The result then follows from Lemma 1.

Consider traffic classes τ_{i1} and $\tau_{k,\ell-j+1}$ for all $\ell \geq j$. We have

$$\begin{aligned}
&\alpha^i - \alpha^k + \gamma\tilde{V}\left((X + A^{k,\ell-j+1})Q, \tilde{h}\right) - \gamma\tilde{V}\left((X + A^{i1})Q, \tilde{h}\right) \\
&= \alpha^i - \alpha^k + \gamma\tilde{V}\left((X + A^{k,\ell-j+1})Q, \tilde{h}\right) - \gamma\tilde{V}\left(XQ, \tilde{h}\right) \\
&> 0,
\end{aligned}$$

where $A^{i1}Q = 0$ and the inequality follows from the fact that $\alpha^i - \alpha^k > 0$ and the fact that the post-decision state value function is non-decreasing in the elements of X. Therefore, (3.16) holds for traffic classes τ_{i1} and $\tau_{k,\ell-j+1}$.

We now aim to show that (3.16) holds for classes $\tau_{i,j+1}$ and $\tau_{k,\ell+1}$ for all $\ell \geq j$. We know that

$$\tilde{V}\left((X + A^{k,\ell+1})Q, \tilde{r}\right) = E\left[V\left((X + A^{k,\ell+1})Q + L, r'\right)\right]$$

and

$$\tilde{V}\left((X + A^{i,j+1})Q, \tilde{r}\right) = E\left[V\left((X + A^{i,j+1})Q + L, r'\right)\right].$$

Therefore, showing that (3.16) holds for classes $\tau_{i,j+1}$ and $\tau_{k,\ell+1}$ is equivalent to showing that

$$\alpha^i + \gamma V\left(X' + A^{k\ell}, r'\right) > \alpha^k + \gamma V\left(X' + A^{ij}, r'\right), \ \forall X', \tag{3.17}$$

where $X' = XQ + L$.[1] We denote the optimal scheduling action used to compute $V(X, r)$ by $Y^*(X)$.[2] Also,

$$\begin{aligned}
V(X, r) &= Q\left(X, r, Y^*(X)\right) \\
&= u(X, Y^*(X)) + \gamma\tilde{V}\left((X - Y^*(X))Q, r\right).
\end{aligned}$$

Given $Y^*(X)$ such that $\sum_{a=1}^{N}\sum_{b=1}^{M} y^{ab,*} = r$, there are two possible cases for the optimal scheduling action $Y^*\left(X + A^{ij}\right)$ corresponding to $V\left(X + A^{ij}, r\right)$:

1. $Y^*\left(X + A^{ij}\right) = Y^*(X)$, i.e., the scheduling action does not change.
2. $Y^*\left(X + A^{ij}\right) = Y^*(X) + A^{ij} - A^{nm}$, i.e., a packet from τ_{ij} is scheduled instead of a packet from some class τ_{nm}.

Similarly, the optimal scheduling action $Y^*\left(X + A^{k\ell}\right)$ corresponding to $V\left(X + A^{k\ell}, r\right)$ has two cases. However, if $Y^*\left(X + A^{ij}\right)$ is case $\eta \in \{1, 2\}$, then $Y^*\left(X + A^{k\ell}\right)$ is restricted to cases $\eta' = 1, \ldots, \eta$. Below, we prove that (3.17) holds in all cases.

[$Y^*\left(X + A^{ij}\right)$ **is case 2]:** We have

$$\begin{aligned}
&\alpha^k + \gamma V\left(X + A^{ij}, r\right) \\
&= \alpha^k + \gamma Q\left(X + A^{ij}, r, Y^*(X) + A^{ij} - A^{nm}\right)
\end{aligned}$$

[1] For notational simplicity, we use X instead of X' and r instead of r' in the rest of the proof of the lemma.

[2] Although $Y^*(X)$ is the optimal scheduling action in traffic state X and rate state r, for notational simplicity, we do not explicitly write the rate state.

Approved for Public Release; Distribution Unlimited.

and

$$\begin{aligned}
&\alpha^i + \gamma V\left(X + A^{k\ell}, r\right) \\
&\geq \alpha^i + \gamma Q\left(X + A^{k\ell}, r, Y^*(X) + A^{k\ell} - A^{nm}\right) \\
&> \alpha^k + \gamma Q\left(X + A^{ij}, r, Y^*(X) + A^{ij} - A^{nm}\right) \\
&= \alpha^k + \gamma V\left(X + A^{ij}, r\right), \qquad (3.18)
\end{aligned}$$

where the first inequality in (3.18) is strict if $Y^*(X + A^{k\ell}) = Y^*(X)$ (case 1) because then $Y^*(X) + A^{k\ell} - A^{nm}$ is a suboptimal action in traffic state $X + A^{k\ell}$. Meanwhile, equality holds in the first inequality in (3.18) if $Y^*(X + A^{k\ell}) = Y^*(X) + A^{k\ell} - A^{nm}$ (case 2). The second inequality in (3.18) follows from the fact that

$$\begin{aligned}
\alpha^i - \alpha^k &> \gamma \begin{bmatrix} Q\left(X + A^{ij}, r, Y^*(X) + A^{ij} - A^{nm}\right) \\ -Q\left(X + A^{k\ell}, r, Y^*(X) + A^{k\ell} - A^{nm}\right) \end{bmatrix} \\
&= \gamma(\alpha^i - \alpha^k).
\end{aligned}$$

Therefore, by Lemma 1, (3.16) holds for traffic classes $\tau_{i,j+1}$ and $\tau_{k,\ell+1}$.

$[Y^*\left(X + A^{ij}\right)$ **is case 1]:** We prove this case by induction on the deadline class. We have already shown that (3.16) holds for traffic classes τ_{i1} and $\tau_{k,\ell-j+1}$ for all $\ell \geq j$. Our induction hypothesis is that (3.16) holds for traffic classes τ_{ij} and $\tau_{k\ell}$ for all $\ell \geq j$.

We have

$$\begin{aligned}
&\alpha^k + \gamma V\left(X + A^{ij}, r\right) \\
&= \alpha^k + \gamma Q\left(X + A^{ij}, r, Y^*(X)\right) \\
&= \alpha^k + \gamma\left[u(X + A^{ij}, Y^*(X)) + \gamma\tilde{V}\left((X + A^{ij} - Y^*(X))Q, r\right)\right]
\end{aligned}$$

and

$$\begin{aligned}
&\alpha^i + \gamma V\left(X + A^{k\ell}, r\right) \\
&= \alpha^i + \gamma Q\left(X + A^{k\ell}, r, Y^*(X)\right) \\
&= \alpha^i + \gamma\left[u(X + A^{k\ell}, Y^*(X)) + \gamma\tilde{V}\left((X + A^{k\ell} - Y^*(X))Q, r\right)\right].
\end{aligned}$$

Since $u(X + A^{ij}, Y^*(X)) = u(X + A^{k\ell}, Y^*(X))$, to prove the result, we must show that

$$\begin{aligned}
&\alpha^i + \gamma^2\tilde{V}\left((X + A^{k\ell} - Y^*(X))Q, r\right) \\
&> \alpha^k + \gamma^2\tilde{V}\left((X + A^{ij} - Y^*(X))Q, r\right)
\end{aligned}.$$

Rewriting this condition, we get

$$\alpha^{i'} + \gamma\tilde{V}\left((X' + A^{k\ell})Q, r\right) > \alpha^{k'} + \gamma\tilde{V}\left((X' + A^{ij})Q, r\right), \qquad (3.19)$$

where $\alpha^{i'} = \alpha^i/\gamma$, $\alpha^{k'} = \alpha^k/\gamma$, and $X' = X - Y^*(X)$. Equation (3.19) is true by the induction hypothesis. Therefore, by Lemma 1, (3.16) holds for traffic classes $\tau_{i,j+1}$ and $\tau_{k,\ell+1}$. □ □

Approved for Public Release; Distribution Unlimited.

3.2.4.3 Structural properties with respect to the dynamics

Consider the two traffic classes τ_{ij} and $\tau_{k\ell}$. If $\alpha^i \leq \alpha^k$ and $d^j \leq d^\ell$ (equalities do not hold at the same time), then we cannot determine the scheduling urgency using Lemma 2. Intuitively, the lower priority packets with earlier deadlines should have higher scheduling urgency only if (i) scheduling them first is unlikely to decrease the number of higher priority packets that get scheduled over time and (ii) scheduling the higher priority packets first will likely result in missing the deadlines of the lower priority packets. Therefore, the scheduling urgency among these traffic classes depends on more than just their absolute priorities and deadlines: it depends on the buffer states and relative priorities of the various traffic classes, the discount factor γ, the rate state r, and the traffic arrival and channel dynamics. We explore this experimentally in Section 3.3.3.

3.3 Results and Discussion

We present our experimental results in this section. We describe the simulation setup in Section 3.3.1. In Section 3.3.2, we compare our approach to EDF, PQ and WFQ in a simple scheduling scenario. In Section 3.3.3, we demonstrate that the optimal policy has a switch-over type structure that depends on various parameters.

3.3.1 Simulation setup

Due to the complexity of value iteration, we make several assumptions on the number of traffic classes and the arrival distribution to make the problem tractable. We assume that traffic is classified into $N = 2$ priority classes and $M = 2$ deadline classes where classes τ_{12} and τ_{22}, which have the latest deadlines, have maximum buffer occupancies of 1 packet, and classes τ_{11} and τ_{21}, which have the earliest deadlines, have maximum buffer occupancies of 2 packets. We assume Bernoulli arrival processes for each class with

$$\Pr(l^{ij} = 1) = \min\{1, \beta \cdot (i + j)/(N + M)\} \tag{3.20}$$

where $\beta \in [0, 2]$. Based on these probabilities, packets with higher priorities and earlier deadlines are less frequent than packets with lower priorities and later deadlines. Additionally, we assume that the set of channel rates is $\mathcal{R} = \{1, 2, 3\}$ packets/slot. We define the load intensity η and the normalized priority-weighted throughput Ψ as:

$$\eta = \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{M} l_t^{ij} \Bigg/ \sum_{t=1}^{T} r_t \text{ and} \tag{3.21}$$

$$\Psi = \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{M} \alpha^i y_t^{ij} \Bigg/ \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{M} \alpha^i l_t^{ij}, \tag{3.22}$$

where T is the total number of timeslots. In words, η is the ratio of the average packet arrivals over time to the average channel rate over time, and Ψ is the ratio of the average priority-weighted throughput over time to the average priority-weighted arrival rate over

Approved for Public Release; Distribution Unlimited.

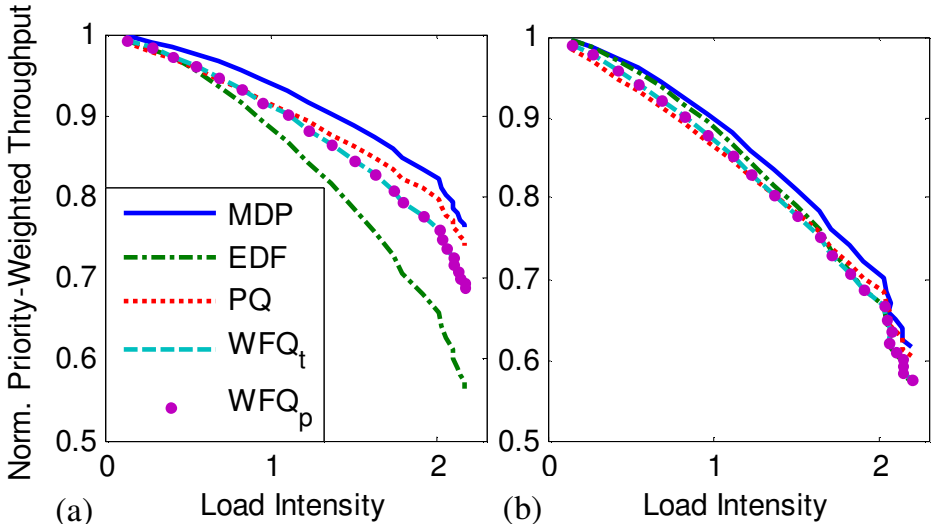


Figure 3: Comparison of normalized priority weighted throughputs for different load intensities. (a) $\alpha^1 = 1$ and $\alpha^2 = 1/4$. (b) $\alpha^1 = 1$ and $\alpha^2 = 3/4$.

time. Several parameters such as the discount factor, priorities of different classes, and rate transition probabilities vary with each simulation, so their values are specified separately in each section.

3.3.2 Comparison to EDF, PQ, and WFQ

In this section, we compare the MDP-based scheduling approach to three simple heuristics: EDF, PQ, and WFQ. Since we do not consider a fluid-based traffic model, we slightly modify WFQ to make it suitable for our problem. Specifically, we use two versions of WFQ for experimental purposes. In the first version, labeled WFQ_t, in each timeslot, a priority class is randomly selected with probability proportional to its priority, and *all* the packets are scheduled from it (from earliest deadline to latest deadline) before any packets from the other priority class, with the total number of scheduled packets limited by the rate constraint. In the second version, labeled WFQ_p, in each timeslot for *each* packet that can be scheduled based on the rate constraint, a priority class is randomly selected with probability proportional to its priority, and a packet with the earliest deadline in that priority class is scheduled. This process is repeated until there are no more packets to be scheduled or the rate constraint is met. Note that, if there are packets in only one of the two priority classes, then packets are scheduled from earliest deadline to latest deadline within that priority class.

In Fig. 3, we plot the normalized priority-weighted throughput, as defined in (3.22), versus load intensity, as defined in (3.21), for our proposed scheduler, PQ, EDF, WFQ_t, and WFQ_p. For illustration, we assume that the rate state has transition probabilities $p_r(\cdot|1) = [0.7, 0.2, 0.1]$, $p_r(\cdot|2) = [0.2, 0.6, 0.2]$, and $p_r(\cdot|3) = [0.2, 0.3, 0.5]$.

In Fig. 3a, we assume that $\alpha^1 = 1$ and $\alpha^2 = 1/4$. In Fig. 3b, we assume that $\alpha^1 = 1$ and $\alpha^2 = 3/4$. It is clear from Fig. 3 that the MDP-based approach is better than PQ, EDF, WFQ_t, and WFQ_p for all load intensities. EDF performs poorly in environments with high load intensities. This is especially evident in Fig. 3a, where there is a large difference in

Approved for Public Release; Distribution Unlimited.

priority between the two priority classes and EDF incurs large penalties when it schedules lower priority packets with earlier deadlines instead of higher priority packets with later deadlines. Both versions of WFQ perform nearly identically and they both underperform the MDP-based scheduling algorithm. The WFQ heuristics perform worst at high traffic load intensities because they allocate too much rate to lower priority packets. In contrast, PQ performs best under high load intensities. It is clear from Fig. 3b that, if the priorities of the traffic classes are similar, then the MDP-based approach does not yield significant benefits over the other heuristics in terms of normalized priority-weighted throughput.

Interestingly, PQ performs within approximately 5% of the MDP approach. This is because there are very few traffic classes and buffer states, and the traffic arrival and channel dynamics do not vary significantly over time (i.e., traffic arrivals are Bernoulli and there are only three channel states); consequently, there are very few cases when it is better to schedule a lower priority packet with earlier deadline (i.e., class τ_{21}) before a higher priority packet with a later deadline (i.e., τ_{12}). In other words, there are few cases when the optimal policy is able to schedule packets that the PQ policy is not able to schedule before their deadlines. When there are more traffic classes, more channel states, and more variability in the arrivals, we believe that the MDP-based approach will perform significantly better than PQ because the optimal scheduling order will be more complex and differ significantly from the order determined by the PQ heuristic. Unfortunately, due to the complexity of value iteration, we are unable to determine the optimal scheduling policy when there are many traffic classes.

In light of these results, we believe that PQ is a good scheduling heuristic to apply for mission critical scheduling in dynamic wireless environments.

3.3.3 Structural properties of the optimal scheduling policy

As we discussed in Section 3.2.4.3, we experimentally study how the urgency between τ_{12} and τ_{21} classes depends on their relative priorities, the discount factor, and the traffic load intensity.

In Fig. 4, we demonstrate how the scheduling policy changes with respect to the discount factor for different traffic states and rate states. The legend at the bottom of Fig. 4 explains how to interpret the different color packets. We assume $\alpha^1 = 1$, $\alpha^2 = 1/2$, $\gamma \in \{0, 0.11, 0.22, \ldots, 0.99\}$, and $p_r(\cdot|1) = p_r(\cdot|2) = p_r(\cdot|3) = [0.2, 0.3, 0.5]$. Note that we consider an i.i.d. rate state to better highlight the structure of the optimal scheduling policy and that Fig. 4 does not include the cases where the rate constraint is large enough to allow all packets in the buffer to be transmitted. We observe that the scheduling policy has a *switch-over* type structure that depends on the discount factor. To illustrate this, each case in Fig. 4 is labeled with a pair (r, γ_{sw}), where r is the rate constraint and γ_{sw} is the switch-over discount factor for that case, which determines the relative urgency between packets from classes τ_{12} and τ_{21}. Specifically, when packets are scheduled with lightly weighted future utilities ($\gamma < \gamma_{\text{sw}}$), the policy schedules packets with higher priorities and later deadlines before the packets with lower priorities and earlier deadlines (i.e., packets from τ_{12} class are more urgent than packets from τ_{21} class). This is because, when the discount factor is low, maximizing the immediate utility is more important than maximizing the future utility. However, when packets are scheduled with heavily weighted future utilities ($\gamma \geq \gamma_{\text{sw}}$), the optimal policy switches to scheduling some packets with lower priorities and

Approved for Public Release; Distribution Unlimited.

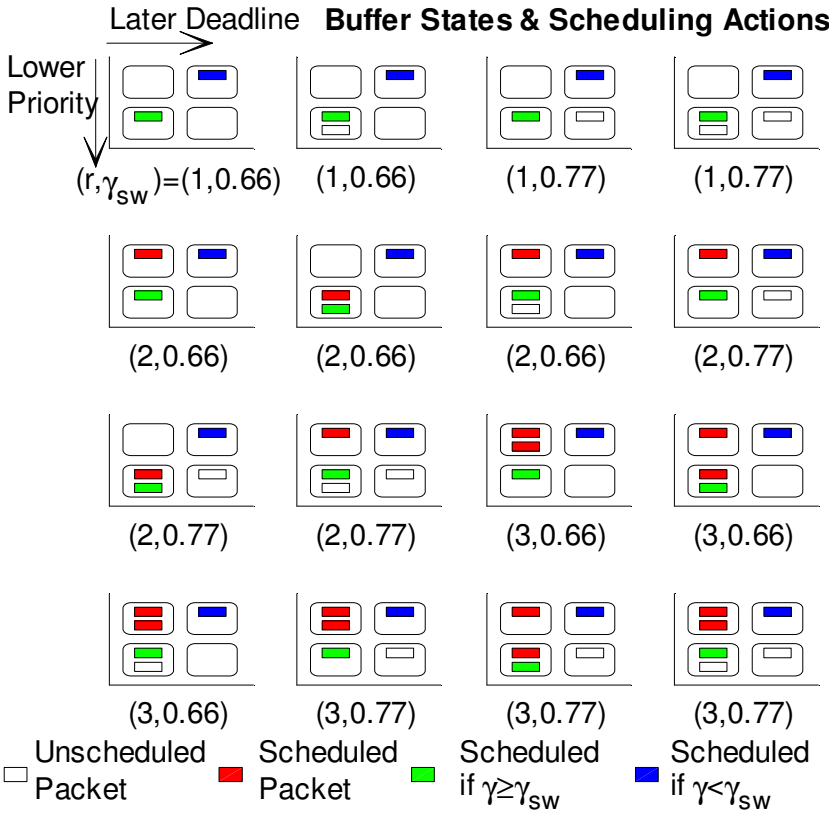


Figure 4: Discount factor values at which optimal packet scheduling switches from τ_{12} class to τ_{21} class for different traffic and rate states. Here, r is the rate constraint, $(\alpha^1, \alpha^2) = (1, \frac{1}{2})$, and γ_{sw} is the switch-over discount factor which determines the urgency between classes τ_{12} and τ_{21}.

Approved for Public Release; Distribution Unlimited.

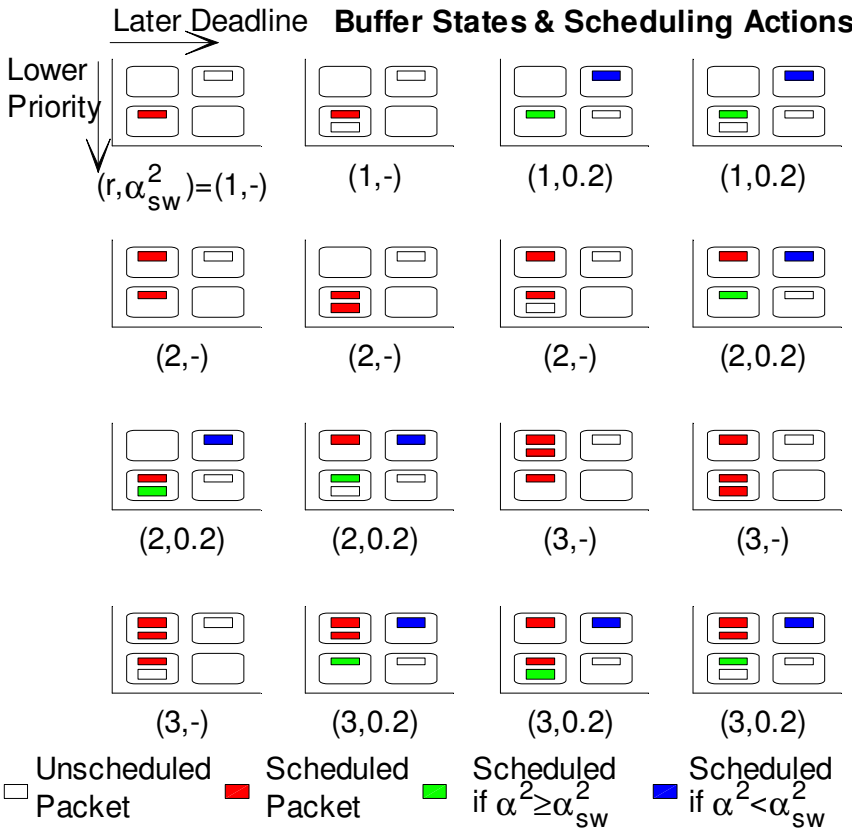


Figure 5: α^2_{sw} values at which packet scheduling switches from τ_{12} class to τ_{21} class for different cases. Here $\gamma = 0.98$, $\alpha^1 = 1$, and "-" indicates that the scheduling action does not change with α^2.

earlier deadlines before the packets with higher priorities and later deadlines (i.e., packets from τ_{21} class become more urgent than packets from τ_{12} class). This is because, when the discount factor is high, the algorithm can see that the higher priority packets that are not scheduled immediately are likely to be scheduled in the next time slot, and therefore are unlikely to miss their deadlines.

In Fig. 5, we show how the scheduling policy changes with respect to the relative priorities between the two classes for $\gamma = 0.98$ and $\alpha^1 = 1$. We experimentally found that the scheduling policy remains unchanged as long as the ratio of priorities between two classes is kept the same (e.g., $(\alpha^1,\ \alpha^2) = (1,\ 0.5)$ and $(\alpha^1,\ \alpha^2) = (4,\ 2)$ result in the same optimal scheduling policy). It turns out that the urgency between two classes depends on the ratio of their priorities, and that the scheduling policy is of switch-over type in this ratio. In particular, when $\alpha^2 \geq \alpha^2_{\text{sw}}$ (i.e., $\alpha^2/\alpha^1 \geq \alpha^2_{\text{sw}}$ since $\alpha^1 = 1$), the policy switches to scheduling packets from τ_{21} class before the packets of τ_{12}. For some cases, we see packets from τ_{21} class scheduled before the packets of τ_{12}, and vice versa, but we also see cases where packets from τ_{21} class are never scheduled before packets from τ_{12} class. This happens because the priority of τ_{21} class is too low compared to that of τ_{12} given the other system variables.

Fig. 6 shows how the load intensity, as defined in (3.21), affects the packet scheduling.

Approved for Public Release; Distribution Unlimited.

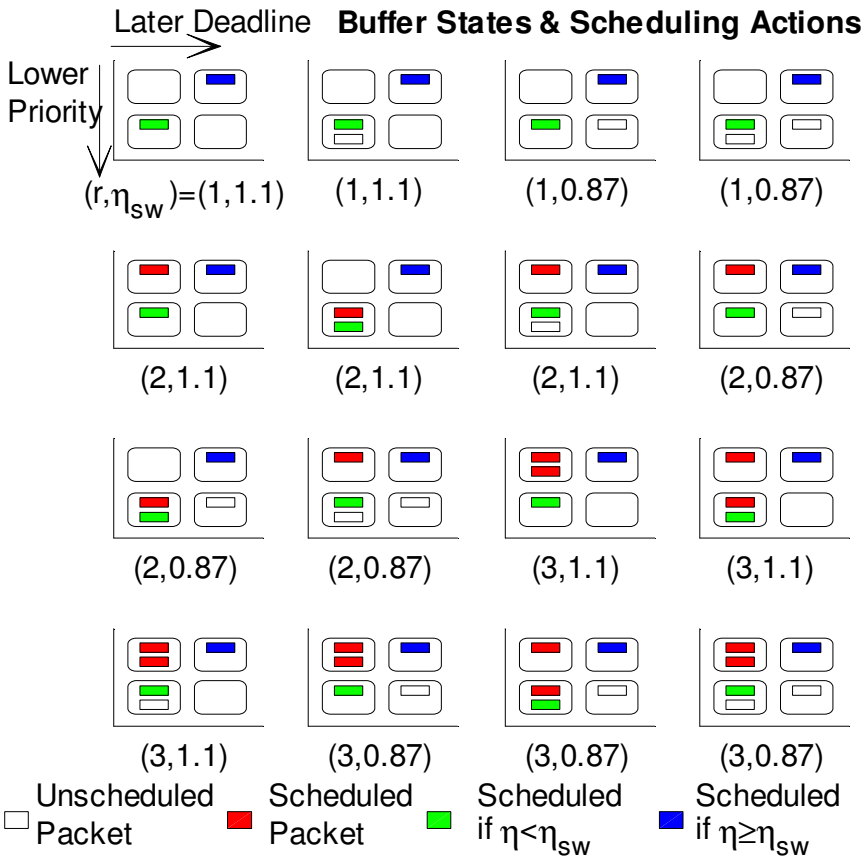


Figure 6: Optimal MDP-based policy schedules earliest deadline class τ_{21} when η is low and the higher priority class τ_{12} when η is high. Here $(\alpha^1, \alpha^2) = (1, \frac{1}{3})$ and $\gamma = 0.98$.

To modulate the load intensity, we modify β defined in (3.20) while keeping the outgoing channel rates the same (i.e., the rate transition probabilities remain the same). We assume that packets from the first priority class are three times more important than the packets from the second priority class. We also assume a discount factor of 0.98. When the load intensity is below a certain value, η_{sw}, lower priority and earlier deadline packets get scheduled before higher priority and later deadline packets. This is because the average channel rate exceeds the average packet arrival rate to a degree that allows the higher priority packets with later deadlines to be scheduled without missing their deadlines (with high probability). For higher load intensities, it is optimal to schedule higher priority packets with later deadlines before lower priority packets with earlier deadlines. If the scheduling urgency is reversed here, lower priority packets will meet their deadlines at the possible expense of higher priority packets missing their deadlines because the average packet arrival rate is higher than the average channel rate. This would result in suboptimal performance.

3.3.4 Discussion

As part of this effort, we formulated the problem of optimal scheduling as an MDP that considers the deadlines and priorities of each packet as well as the arrivals and channel

Approved for Public Release; Distribution Unlimited.

dynamics. The objective of the MDP is to maximize the congested node's long-run utility (i.e., priority-weighted throughput) subject to instantaneous transmission rate constraints. We showed theoretically that a class has a higher scheduling urgency than another if it has a higher priority and an earlier deadline, the same deadline and a higher priority, or the same priority and an earlier deadline. In order to understand the scheduling urgency between a higher priority class with a later deadline, and a lower priority class with an earlier deadline, we experimentally investigated the structure of the optimal scheduling policy. Our experimental results demonstrated that the optimal policy switches from scheduling a higher priority class with a later deadline before a lower priority class with an earlier deadline, to the opposite order, when the discount factor exceeds a threshold, when the ratio of the priorities of the two classes exceeds a threshold, or when the load intensity is below a threshold. These thresholds depend on the dynamics of the system. Our results also show that the MDP based approach outperforms PQ, EDF and WFQ over a wide range of load intensities; however, we are unfortunately unable to simulate a case where the proposed solution significantly outperforms the PQ heuristic. This is because, given the complexity of value iteration, we are only able to consider a limited number of traffic classes and simplistic channel and arrival dynamics for which the PQ heuristic is near optimal.

Approved for Public Release; Distribution Unlimited.

4. *Delay-Sensitive CSMA/CA Scheduling*

4.1 Introduction

Distributed MAC protocols allow multiple users to access a shared channel without the help of a centralized controller and, consequently, are robust to node failures. This makes them suitable for ad hoc networks in general, and airborne networks in particular. However, such protocols make it very difficult to provide the necessary Quality of Service (QoS) to heterogeneous users with different resource requirements and application constraints. This is because users must rely on local information to make scheduling decisions, yet their decisions are coupled by the limited network resources, i.e., a user's decisions not only affect its own performance, but also affect the other users.

In this section, we consider the problem of energy-efficient scheduling of delay-sensitive data over fading channels using a CSMA/CA-based distributed MAC protocol. Although conventional aircraft are not energy constrained, using the minimum required transmission power to achieve a particular delay constraint is still beneficial because it generates less interference at distant nodes and it may reduce the probability of intercept and probability of detection of transmitted signals. On the other hand, energy-efficiency is important in the context of low cost attritable small UAS. For example, measurements in [33] suggest that communications can account for up to 20% of a multi-rotor's energy consumption and, intuitively, this fraction should be even higher for small fixed-wing drones, which require less energy for flight than multi-rotors do. Thus, more energy-efficient communications directly translates to longer flight times for small UAS.

There is a lot of prior research on multi-user scheduling. In [34,35], the authors formulate the problem of multi-user uplink video streaming as a Markov decision process (MDP) with the objective of maximizing the sum of utilities across the users. They consider a time division multiple access (TDMA)-based medium access control (MAC), which divides each time slot into transmission opportunities with durations proportional to the number of packets that each user wants to transmit. While these studies consider delay-sensitive traffic, they disregard energy consumption. In [36], the authors consider the problem of energy-efficient delay-sensitive multi-user uplink scheduling in an IEEE 802.16 (WiMAX) network. They formulate the M user problem as an MDP and decompose it into $M + 1$ sub-problems: M user problems, in which each user selects the number of packets that it wants to transmit, and one base station problem, in which the base station allocates the channel using TDMA to the user that wants to transmit the most packets. In this project, we consider a similar problem as [36], but with a CSMA/CA-based MAC.

Enabling users to efficiently trade off energy and delay using a conventional CSMA/CA-

Approved for Public Release; Distribution Unlimited.

based MAC, exemplified by the IEEE 802.11 Distributed Coordination Function (DCF) [37], is a challenging problem. This is because CSMA/CA typically provides users equal channel access probabilities, thereby ignoring that some users may need to transmit data with higher urgency than other users at different times. Although many techniques have been proposed to provide users with differentiated channel access probabilities using CSMA/CA (e.g., assigning users different minimum backoff window sizes [38, 39], backoff schedules [38, 40], maximum backoff stages [38], and/or Inter-Frame Spacing values [40]), this is often done in a static way such that a specific user or flow always has the same channel access probability. MAC in motion [41] provides users different channel access probabilities in vehicular networks based on their distance to roadside access points. Collaborative Virtual Sensing [30] provides users different channel access probabilities over time; however, it requires users to overhear significant information from other users in order to set their backoff counters. Notably, both [41] and [30] focus on network throughput rather than energy and delay.

Our contributions are threefold:

1. We formulate the energy-efficient delay-sensitive multi-user scheduling problem as an MDP and show that it is intractable. We propose to solve it by decomposing it into multiple (coupled) single-user problems, similar to [36].

2. We propose a reinforcement learning algorithm to solve the single-user problems online, thereby enabling users to minimize their energy consumption subject to their delay constraints, even though the channel, traffic, and multi-user dynamics are unknown a priori. The proposed learning algorithm, which we adapt from our prior work on single-user scheduling [25], converges significantly faster than the state-of-the-art learning algorithm in [36].

3. Instead of relying on a TDMA-based MAC as in [34–36], we propose a fully distributed rate-adaptive CSMA/CA protocol. Unlike traditional CSMA/CA, which provides users equal channel access probabilities, the proposed solution provides higher access probabilities to users that most urgently need to transmit their data by assigning them smaller congestion windows. The MAC protocol works in tandem with the rate adaptation algorithm at the physical layer to help users minimize their energy consumption subject to their delay constraints.

The remainder of this section is organized as follows. In Section 4.2, we introduce the system model. In Section 4.2.4, we formulate the multi-user scheduling problem as an MDP and decompose it into multiple tractable single-user problems. In Section 4.2.5, we describe how to solve the single-user problems online using reinforcement learning. In Section 4.2.6, we propose a rate-adaptive CSMA/CA protocol. In Section 4.3, we present our simulation results. We conclude in Section 4.3.3.

4.2 Methods, Assumptions and Procedures

We consider the problem of multi-user scheduling in a CSMA/CA-based network with M users indexed by $i \in \{1, ..., M\}$. We assume that time is slotted into discrete-time intervals

Approved for Public Release; Distribution Unlimited.

with equal duration Δt seconds and that only one user is allowed to transmit in each time slot. We let $n \in \mathbb{N}$ denote the time slot index. Since we consider CSMA/CA, users are scheduled in a distributed manner. We consider a *block fading* channel where the *channel gain* $h_i^n \in \mathcal{H}$ experienced by user i remains constant within each time slot, but can vary between time slots. As in [25,34,36,42], we assume that the set of channel gains (hereafter, channel states) $\mathcal{H}$ is discrete and finite and that the sequence of channel states $\{h_i^n \in \mathcal{H} : n = 0, 1, \ldots\}$ can be modeled as a Markov chain with transition probability function $p_i^h(h_i^{n+1}|h_i^n)$. We also assume that the users' channel states are statistically independent and that the channel transition probabilities are *unknown* a priori.

4.2.1 Physical Layer Model

The physical layer is assumed to be a single-input single-output system designed to handle quadrature amplitude modulation (QAM) square constellations, with fixed symbol rate $1/T_s$ symbols/s where T_s is the symbol duration. Let $\mathcal{M}$ denote the set of QAM constellations available to each user and let $\beta_i \in \mathcal{M}$ denote the number of bits per symbol in the ith user's selected QAM constellation. Under β_i, the ith user transmits at the physical layer rate β_i/T_s bits/s. We let $\beta^{\max}$ denote the maximum number of bits per symbol in $\mathcal{M}$.

Given the number of bits per symbol β_i and channel state h_i, the *transmission power* P_i required to meet a target *bit-error rate*, BER_i, is well approximated by [43]:

$$P(h_i, \beta_i; BER_i) = \frac{N_0(2^{\beta_i}-1)}{3T_s h_i} \left[Q^{-1}\left(\left(1 - 2^{-\beta_i/2}\right)^{-1} \frac{\beta_i}{4} BER_i \right) \right]^2 \text{ Watts}, \tag{4.1}$$

where N_0 is the noise power spectral density (Watts/Hz) and $Q^{-1}(\cdot)$ is the inverse of

$$Q(y) \triangleq \frac{1}{\sqrt{1/2\pi}} \int_{z=y}^{\infty} \exp(-z^2/2) dz = \frac{1}{2} \text{erfc}\left(\frac{y}{\sqrt{2}} \right). \tag{4.2}$$

Note that the transmission power $P(h_i, \beta_i; BER_i)$ is convex and increasing in the transmission rate β_i, decreasing in the channel state h_i, and decreasing in BER_i.

Using modulation β_i, and assuming that application layer packets have a fixed size of L bits, it is possible to transmit

$$r(\beta_i; t) = \lfloor \beta_i t / T_s L \rfloor \text{ packets} \tag{4.3}$$

in t seconds, where $\lfloor x \rfloor$ denotes the floor of x. The total transmission energy consumed by user i over t seconds using modulation β_i can be calculated as

$$e_i(h_i, \beta_i; t) = P(h_i, \beta_i; BER_i) \cdot t \text{ Joules}. \tag{4.4}$$

Note that, in general, the number of packets that are actually decodable by the receiver may be less than $r(\beta_i; t)$ due to transmission errors. To capture this, packet losses can be integrated into the physical layer model as described in our prior work [25]; however, to simplify the exposition in this project, we assume that the target bit-error rate is sufficiently small such that the packet error rate is negligible.

Approved for Public Release; Distribution Unlimited.

4.2.2 Data Link Layer Model

We now present our data link layer model, which includes the MAC protocol and the traffic buffer. Note that we wait until Section 4.2.6 to describe the MAC protocol in full detail. We consider a rate-adaptive CSMA/CA-based MAC. We assume that each time slot is divided into two phases: a *contention phase*, during which users contend for channel access; and a *transmission phase*, during which one user transmits while the others keep silent. In time slot n, the contention phase has length T^n_{MAC}, $0 < T^n_{\text{MAC}} \leq \Delta t$, and the transmission phase has length $T^n_{\text{TX}} = \Delta t - T^n_{\text{MAC}}$, where Δt is the time slot duration. Let $x^n_i \in \{0, 1\}$ be an indicator variable that is set to 1 if user i gets access to the channel in time slot n and is set to 0 otherwise. Since at most one user can access the channel in time slot n, the components of the *channel access indicator vector* $x^n = (x^n_1, \ldots, x^n_M)$ must satisfy the *channel access constraint* $\sum_{i=1}^{M} x^n_i \leq 1$. Note that, through the rate-adaptive CSMA/CA protocol proposed in Section 4.2.6, the contention time T^n_{MAC}, transmission time T^n_{TX}, and channel access indicator vector x^n are all random variables that depend on the users' selected modulation schemes in time slot n.

We let $b^n_i \in \{0, 1, \ldots\} = \mathcal{B}$ denote the ith user's buffer backlog (in packets) at the beginning of time slot n and let u^n_i denote the number of packets that user i actually transmits in time slot n. If user i does not get access to the channel, i.e., $x^n_i = 0$, then $u^n_i = 0$. On the other hand, if user i gets access to the channel, i.e., $x^n_i = 1$, then $u^n_i = \min\{r(\beta^n_i; T^n_{\text{TX}}), b^n_i\}$, where β^n_i is its selected modulation scheme, T^n_{TX} is the transmission phase duration, and $r(\beta^n_i; T^n_{\text{TX}})$ is defined in (4.3). It follows that u^n_i can be compactly represented as

$$u^n_i = x^n_i \min\{r(\beta^n_i; T^n_{\text{TX}}), b^n_i\}. \tag{4.5}$$

The ith user's buffer state evolves as follows:

$$b^{n+1}_i = b^n_i - u^n_i + l^n_i, \tag{4.6}$$

where l^n_i is the number of packets that the application layer injects into the transmission buffer at the end of time slot n. We assume that the arrival process $\{l^n_i : n = 0, 1, ...\}$ is a sequence of i.i.d. random variables with l^n_i distributed according to the *packet arrival distribution* $p^l_i(l_i)$ (however, our framework can also be applied to correlated traffic where the sequence of arrivals is a Markov chain). Furthermore, we assume that the users' packet arrivals are statistically independent and that their packet arrival distributions are *unknown* a priori.

As described in the introduction of this section, we assume that every user has its own average packet delay constraint. By Little's theorem [44], the average delay is proportional to the average buffer occupancy. Hence, we may use the average buffer state as a proxy for the delay. As such, we define the *delay cost*

$$d_i(b_i) = b_i, \tag{4.7}$$

which we will also refer to as the *holding cost*.

4.2.3 Summary of the System's Operation

We now describe how the system operates. At the beginning of each time slot n, each user $i \in \{1, 2, \ldots, M\}$ observes its state $s^n_i = (b^n_i, h^n_i) \in \mathcal{S} = \mathcal{B} \times \mathcal{H}$, which comprises its buffer

Approved for Public Release; Distribution Unlimited.

state b_i^n and channel state h_i^n. Then, based on its state, each user i determines its modulation scheme β_i^n as described later in Section 4.2.5. Subsequently, the users contend for channel access using the CSMA/CA protocol defined later in Section 4.2.6. When one user gains access to the channel, the other users remain silent. Whether or not it gets channel access, user i transmits u_i^n packets as defined in (4.5), incurs a delay cost $d_i(b_i)$ as defined in (4.7), and consumes transmission energy $x_i^n e_i(h_i^n, \beta_i^n; T_{\text{TX}}^n)$, where $e_i(h_i, \beta_i; t)$ is defined in (4.4) and x_i^n is its channel access indicator. Finally, at the end of each time slot n, user i experiences $l_i^n \sim p_i^l(l_i)$ new packet arrivals, its next buffer state b_i^{n+1} is determined according to (4.6), and it transitions to a new channel state $h_i^{n+1} \sim p_i^h(h_i|h_i^n)$. Based on the above description, it is clear that users are coupled through the shared wireless channel: indeed, users who do not get channel access cannot drain their buffers, so they incur higher future delay costs.

4.2.4 Problem Formulation

4.2.4.1 Multi-User Problem Formulation

In this section, we formulate the energy-efficient delay-sensitive multi-user scheduling problem. The objective of the scheduling problem is to minimize each user's *infinite horizon discounted energy costs* subject to a constraint on each user's *infinite horizon discounted delay*. We note that the use of *discounted* costs for energy management problems has been thoroughly justified in [45]. The ith user's discounted energy and delay costs can be expressed as

$$E_i = \mathbb{E}\left[\sum_{n=0}^{\infty} (\gamma)^n x_i^n e_i(h_i^n, \beta_i^n; T_{\text{TX}}^n)\right] \text{ and} \tag{4.8}$$

$$D_i = \mathbb{E}\left[\sum_{n=0}^{\infty} (\gamma)^n d_i(b_i^n)\right], \tag{4.9}$$

respectively, where $\mathbb{E}[\cdot]$ denotes an expectation over the sequence of user i's buffer and channel states; $\gamma \in [0, 1)$ is the *discount factor*; and $(\gamma)^n$ denotes the discount factor to the nth power. In words, the ith user's discounted energy cost (delay) is the expected cumulative sum of energy (delay) accrued from now to the end of time, where the energy (delay) incurred n time steps in the future is discounted by $(\gamma)^n$.

Stated formally, the objective of the multi-user scheduling problem is to determine the users' modulation schemes in each time slot in order to solve the following optimization:

$$\text{Minimize } E_i \text{ subject to } D_i \leq \delta_i, \ \forall i \in \{1, \ldots, M\}, \tag{4.10}$$

where δ_i is the ith user's delay constraint. Importantly, (4.10) effectively maximizes bits/Joule (i.e., energy efficiency) because it minimizes the energy required to transmit sufficient data to meet the delay constraint. We note that (4.10) can be formulated as a constrained MDP (CMDP) with M state variables $s = (s_1, \ldots, s_M)$ and M actions $\beta = (\beta_1, \ldots, \beta_M)$. It was shown in [46] that a constrained MDP can be reformulated as an unconstrained MDP. Therefore, in principle, the optimal solution to (4.10) can be computed using the well-known value iteration algorithm [47]; however, this is impractical for three reasons. First, the complexity of solving an MDP is proportional to the cardinality of its state-space $\mathcal{S}$, which increases

Approved for Public Release; Distribution Unlimited.

exponentially with the number of users M. Hence, even for a moderate number of users, it is impractical to compute the optimal solution. Second, in practice, each users' traffic arrival and channel state transition dynamics are unknown a priori. Consequently, even if we ignore the computational complexity, we cannot directly apply value iteration. Finally, even if we are able to compute the optimal solution, it would require each user to know the other users' states in each time slot. Unfortunately, exchanging this information would incur significant communication overheads.

In the next subsection, we propose to approximately solve (4.10) by decomposing it into M single-user problems. Each single-user problem depends on only one user's state information and has solution complexity that is independent of the number of users. In Section 4.2.5, we discuss how the single-user problems can be solved online using reinforcement learning to deal with the fact that the traffic, channel, and multi-user dynamics are unknown a priori.

4.2.4.2 Multi-User Problem Decomposition

In order to decompose the multi-user problem into M single-user problems, we make the following approximation:

Definition 2 (Single-user approximation)**.** *Each user operates under the assumption that it is the only user in the network and that it has the entire time slot available to transmit its data.*

From the ith user's perspective, this approximation implies that the transmission phase duration $T_{\text{TX}}^n = \Delta t$, its channel access indicator $x_i^n = 1$, and its transmission rate is equal to $\min\{r(\beta_i^n; \Delta t), b_i^n\}$. Note that this approximation represents how each user *models* its environment, but not how the environment actually behaves, i.e., users' are still coupled through the shared wireless channel.

We now formulate the local optimizations applied by each user under the single-user approximation. The ith user aims to determine a *policy* π_i, which maps its local states to its local actions [i.e., $\beta_i = \pi_i(b_i, h_i)$], and minimizes its discounted energy consumption subject to its discounted delay constraint. When user i follows a policy π_i, its discounted energy and delay costs can be expressed as

$$E_i^{\pi_i} = \mathbb{E}\left[\sum_{n=0}^{\infty} (\gamma)^n e_i(h_i^n, \beta_i^n; \Delta t)\right] \text{ and} \tag{4.11}$$

$$D_i^{\pi_i} = \mathbb{E}\left[\sum_{n=0}^{\infty} (\gamma)^n d_i(b_i^n)\right], \tag{4.12}$$

respectively, where its transmission energy $e_i(h_i^n, \beta_i^n; \Delta t)$ is defined in (4.4); its modulation scheme in each time slot is selected by following its policy π_i; and $E[\cdot]$ denotes an expectation over the sequence of its local buffer and channel states. Note that, in accordance with the single-user approximation, user i's discounted energy defined in (4.11) is determined assuming that it is the only user in the network and that it has the entire time slot available to transmit its data. Now, stated formally, the objective of the constrained single-user scheduling problem is to determine the optimal policy π_i^*, which is the solution to the following

Approved for Public Release; Distribution Unlimited.

problem:

$$\text{Minimize } E_i^{\pi_i} \text{ subject to } D_i^{\pi_i} \leq \delta_i, \tag{4.13}$$

where δ_i is the ith user's delay constraint and $E_i^{\pi_i}$ and $D_i^{\pi_i}$ are its discounted energy and delay costs under the single-user approximation, respectively.

It was shown in [46] that a constrained CMDP like the one defined in (4.13) can be reformulated as an unconstrained MDP by introducing a Lagrange multiplier associated with the delay constraint. We define user i's Lagrangian cost function as

$$c_i^{\lambda_i}([b_i, h_i], \beta_i) = e_i(h_i, \beta_i; \Delta t) + \lambda_i d_i(b_i). \tag{4.14}$$

For a fixed λ_i, the ith user's unconstrained problem's objective is to minimize its *infinite horizon discounted Lagrangian cost*:

$$L_i^{\lambda_i} = \mathbb{E}\left[\sum_{n=0}^{\infty} (\gamma)^n c_i^{\lambda_i}([b_i, h_i], \beta_i)\right]. \tag{4.15}$$

The optimal solution to (4.15) satisfies the following Bellman equation:

$$V_i^{\lambda_i,*}(b_i, h_i) = \min_{\beta_i \in \mathcal{M}} \left\{ \begin{array}{l} c_i^{\lambda_i}([b_i, h_i], \beta_i) + \\ \gamma \mathbb{E}_{l_i, h'} \left[V_i^{\lambda_i *}([b_i - r(\beta_i; \Delta t)]^+ + l_i, h_i') \right] \end{array} \right\}, \tag{4.16}$$

where $V_i^{\lambda_i,*}(b_i, h_i)$ is the ith user's *optimal state-value function*, $r(\beta_i^n; \Delta t)$ is defined in (4.3), $[x]^+ = \max\{x, 0\}$, and the expectation is taken over the arrival distribution and channel transition probabilities. Given $V_i^{\lambda_i,*}$, user i's optimal policy $\pi_i^{\lambda_i,*}$ can be determined by taking the argument that minimizes the right-hand side of (4.16).

In practice, the arrival distribution and channel transition probabilities are *unknown* a priori. Moreover, the multi-user dynamics will result in users transmitting less packets than they expect under the single-user approximation. Consequently, V_i^* and π_i^* cannot be computed using value iteration; instead, they must be learned online based on experience.

4.2.5 Learning the Optimal Policy

In this section, we describe an algorithm that enables each user to learn its optimal state value function V_i^* and optimal policy π_i^* online (hereafter, we drop the Lagrange multiplier from the notation for brevity). The algorithm is based on the so-called *post-decision state learning algorithm with virtual experience* proposed in our prior work [25], which we used to solve a single-user point-to-point scheduling problem. The remainder of this section is organized as follows. In Section 4.2.5.1, we review the *post-decision state* concept (which we first introduced in Section 3), define a new value function based on the post-decision state, and present an algorithm to learn this new value function online, similar to the learning algorithm in [36]. In Section 4.2.5.2, we enhance the post-decision state learning algorithm by introducing *virtual experience updates*, which dramatically improve the learning algorithm's convergence rate.

Approved for Public Release; Distribution Unlimited.

4.2.5.1 The Post-Decision State Learning Algorithm

A *post-decision state* (PDS) is an intermediate state that occurs after the packet transmission takes place, but before the packet arrivals and next channel state are realized. For user $i \in \{1, \ldots, M\}$, we denote the PDS as $\tilde{s}_i \in \mathcal{S}$, where the set of possible PDSs is the same as the set of possible states. The ith user's PDS in time slot n is related to its state $s_i^n = (b_i^n, h_i^n)$ and action β_i^n as follows:

$$\tilde{s}_i^n = (\tilde{b}_i^n, \tilde{h}_i^n) = ([b_i^n - r(\beta_i^n; \Delta t)]^+, h_i^n). \tag{4.17}$$

In words, the buffer's PDS $\tilde{b}_i^n = [b_i^n - r(\beta_i^n; \Delta t)]^+$ characterizes the buffer state *after* the packets are transmitted, but *before* new packets arrive. We optimistically use $[b_i^n - r(\beta_i^n; \Delta t)]^+$ as the post-decision buffer state instead of $[b_i^n - x_i^n r(\beta_i^n; T_{\text{TX}}^n)]^+$ because the ith user does not know if it will get access to the channel, or how long the transmission phase will be, until after it selects its action and contends for channel access as described in Section 4.2.6. This modeling assumption is in accordance with the single-user approximation described in Section 4.2.4.2. Meanwhile, the channel's PDS $\tilde{h}_i^n = h_i^n$ is the same as the channel state at time n.

Before we can describe the PDS learning algorithm, we need to define the *PDS value function*, which is a value function defined over the PDSs. The ith user's optimal PDS value function, denoted by $\tilde{V}_i^*$, can be expressed as a function of the optimal state-value function V_i^*, and vice-versa:

$$\tilde{V}_i^*(\tilde{b}_i, \tilde{h}_i) = \sum_{l, h_i'} p_i^l(l_i) p_i^h(h_i' | \tilde{h}_i) V_i^*(\tilde{b}_i + l_i, h_i'), \tag{4.18}$$

$$V_i^*(b_i, h_i) = \min_{\beta_i \in \mathcal{M}} \left\{ \begin{array}{l} c_i([b_i, h_i], \beta_i) + \\ \gamma \tilde{V}_i^*([b_i - r(\beta_i; \Delta t)]^+, h_i) \end{array} \right\}, \tag{4.19}$$

Given the optimal PDS value function, the optimal policy $\pi_i^*(b_i, h_i)$ can be computed by taking the argument that minimizes the right-hand side of (4.19).

PDS learning is a stochastic iterative algorithm that each user can deploy to learn its optimal PDS value function $\tilde{V}_i^*$ and policy π_i^* through its interactions with the environment. The algorithm is summarized in Table 2. Central to PDS learning is the simple update step in (4.20), which is performed at the end of each time slot based on user i's experience tuple $\sigma_i^n = (s_i^n, \beta_i^n, \tilde{s}_i^n, s_i^{n+1})$ in time slot n, where $s_i^n = (b_i^n, h_i^n)$ is its state; β_i^n is its modulation scheme; $\tilde{s}_i^n$ is its post-decision state as defined in (4.17); and $s_i^{n+1} = (b_i^{n+1}, h_i^{n+1})$ is its resulting state in time slot $n+1$. The parameter α_i^n in (4.20) is a learning rate that satisfies the *stochastic approximation conditions* $\sum_{n=0}^{\infty} \alpha_i^n = \infty$ and $\sum_{n=0}^{\infty} (\alpha_i^n)^2 < \infty$. The optimal value of the Lagrange multiplier λ_i, which depends on the delay constraint δ_i, is learned online using stochastic subgradients as shown in (4.21), where Λ projects λ onto $[0, \lambda_{\max}]$. In (4.21), ε_i^n is a time-varying learning rate, which must satisfy the same stochastic approximation conditions as α_i^n, and $d(b_i^n)$ is the delay cost. The following additional conditions must be satisfied by ε_i^n and α_i^n to ensure convergence of (4.21) to the optimal Lagrange multiplier λ_i^*: $\sum_{n=0}^{\infty} (\alpha_i^n + \varepsilon_i^n) < \infty$ and $\lim_{n \to \infty} \varepsilon_i^n / \alpha_i^n \to 0$.

Approved for Public Release; Distribution Unlimited.

Table 2: Post-decision state learning algorithm at user i.

1.	At time $n = 0$, initialize $\tilde{V}_i^0$ and V_i^0 arbitrarily (e.g., to 0).
2.	At time n, given the state $s_i^n = (b_i^n, h_i^n)$, take the greedy action β_i^n that minimizes the right-hand side of (4.19) using $\tilde{V}_i^n$ in place of $\tilde{V}_i^*$.
3.	Record the experience tuple $\sigma_i^n = (s_i^n, \beta_i^n, \tilde{s}_i^n, s_i^{n+1})$.
4.	Compute $V_i^n(s_i^{n+1}) = V_i^n([b_i^n - r(\beta_i^n; \Delta t)]^+ + l_i^n, h_i^{n+1})$ using (4.19) with V_i^n and $\tilde{V}_i^n$ in place of V_i^* and $\tilde{V}_i^*$, resp.
5.	Update the PDS value function using the result of step 4: $$\tilde{V}_i^{n+1}(\tilde{s}_i^n) \leftarrow (1 - \alpha_i^n)\tilde{V}_i^n(\tilde{s}_i^n) + \alpha_i^n V_i^n(s_i') \quad (4.20)$$
6.	Update the Lagrange multiplier: $$\lambda_i^{n+1} = \Lambda\left[\lambda_i^n + \varepsilon_i^n(d(b_i^n) - \delta_i)\right] \quad (4.21)$$
7.	**Repeat:** $n \leftarrow n + 1$. Go to step 2.

4.2.5.2 Virtual Experience Learning

The PDS learning algorithm only updates one PDS in each time slot. However, it is possible to update multiple PDSs in each time slot in order to accelerate the learning rate and improve run-time performance. The key idea is that the new traffic arrivals l_i^n and next channel state h_i^{n+1} are statistically independent of the buffer state b_i^n and action β_i^n. Therefore, we can update any PDSs with the same post-decision channel state $\tilde{h}_i^n$, but different post-decision buffer states $\tilde{b}_i^n$, for the current observations of l_i^n and h_i^{n+1}. That is, given user i's experience tuple in time slot n, i.e., $\sigma_i^n = (s_i^n, \beta_i^n, \tilde{s}_i^n, s_i^{n+1})$, steps 4 and 5 of the PDS learning algorithm in Table 2 can be applied to any *virtual experience tuple* in the set

$$\mathcal{V}(\sigma_i^n) = \left\{(s_i^n, \beta_i^n, [\tilde{b}_i, \tilde{h}_i^n], [\tilde{b}_i + l_i^n, h_i^{n+1}]) | \forall \tilde{b}_i \in \mathcal{B}_i\right\}, \quad (4.22)$$

where $\hat{s}_i = (\tilde{b}_i, \tilde{h}_i^n)$ and $\hat{s}_i' = (\tilde{b}_i + l_i^n, h_i^{n+1})$ denote the virtual PDS and the virtual next state, respectively. Virtual experience learning's complexity increases linearly with the number of virtual experience updates applied in each time slot; however, the update in (4.20) can be applied to virtual experience tuples in parallel to reduce execution time.

4.2.6 Rate-adaptive CSMA/CA Protocol

For our rate-adaptive CSMA/CA protocol, we adopt the IEEE 802.11 DCF with RTS/CTS handshaking, but with two modifications. First, unlike the IEEE 802.11 DCF, which provides users equal channel access probabilities, we determine the *congestion windows* (CWs) so

Approved for Public Release; Distribution Unlimited.

users that desire higher transmission rates have higher channel access probabilities. In this way, when coupled with the proposed transmission scheduling algorithm, the MAC protocol supports the objective of minimizing each users' energy consumption subject to their delay constraints. This is because, as noted in [36], a user will want to transmit at a higher rate (i) if it has a large queue backlog, and is therefore likely to violate its delay constraint, or (ii) if it has a good channel, and can therefore transmit with low power. Second, unlike the IEEE 802.11 DCF, where users freeze their backoff counters when another user grabs the channel and resume them once the channel is free, we assume that users reset their backoff counters at the end of each time slot. In this way, a user's backoff counter can be updated to reflect its current state. Additionally, limiting the time that low rate users spend on the channel helps mitigate an anomaly in CSMA/CA that results in low rate users degrading the performance of all users [48].

The rate-adaptive CSMA/CA protocol works as follows. At the beginning of each time slot in which it wants to transmit, the ith user sets its backoff randomly and uniformly in the range $[0, CW_{\min}(\beta_i^n) - 1]$, where $CW_{\min}(\beta_i^n)$ is the minimum CW size, which depends on the user's selected modulation β_i^n. Although $CW_{\min}(\cdot)$ may take many functional forms, for illustration, we assume that it is defined as follows:

$$CW_{\min}(\beta_i) = \left\lfloor A \cdot 2^{\beta_{\max} - \beta_i} \right\rfloor, \tag{4.23}$$

where A is a positive real number and $\beta_{\max}$ is the maximum number of bits per symbol supported by the physical layer. In (4.23), the minimum congestion window is defined such that users who want to transmit at higher (lower) rates will have smaller (larger) congestion windows, resulting in higher (lower) channel access probabilities. Lastly, if a user's RTS packet is not successfully received by the access point (i.e., there is a collision), then the user's congestion window is doubled, up to a maximum value $CW_{\max} = A \cdot 2^{\beta_{\max}}$.

4.3 Results and Discussion

In this section, we demonstrate through MATLAB simulations that (i) the proposed virtual experience learning algorithm significantly outperforms the state-of-the-art PDS learning algorithm in [36] and that (ii) the proposed rate-adaptive CSMA/CA protocol enables users to achieve significantly better energy and delay performance than the IEEE 802.11 DCF. In our simulations, the ith user's signal-to-noise ratio (SNR) in time slot n is computed as $\text{SNR}_i^n = h_i^n P_i^n / N_0 W$, where h_i^n is the channel gain at the receiver, P_i^n is the transmission power, N_0 is the noise power spectral density, and W is the bandwidth. We consider Poisson arrivals for each user; however, we assume that the arrival distribution and channel transition probabilities are not known to the users a priori. We assume that users can select among 5 different modulations, namely, BPSK, QPSK, 16-QAM, 64-QAM, and 256-QAM, and that they adapt their transmission powers to maintain a packet loss rate below 1% for packets with size 5000 bits. Table 3 summarizes the key parameters used in our simulations.

Approved for Public Release; Distribution Unlimited.

Table 3: Simulation parameters.

Parameter	**Value**
Average arrival rate	Variable: 1-3 packets/time slot
Bits per symbol $\beta \in \mathcal{M}$	{1, 2, 4, 6, 8}
Channel states $h \in \mathcal{H}$	{-18.82, -13.79, -11.23, -9.37, -7.80, -6.30, -4.68, -2.08} dB
Discount factor γ	0.98
Delay constraint δ_i	Variable: 3.25 – 12 packets
Noise power spectral density N_0	2×10^{-11} W/Hz
Packet loss rates PLR	1%
Packet size L	5000 bits
Symbol rate $1/T_s$	500×10^3 symbols/s
Time slot duration Δt	10 ms
Number of users M	3 or 10
Minimum CW size A	2
Slot time, DIFS, SIFS	9 μs, 34 μs, 16 μs

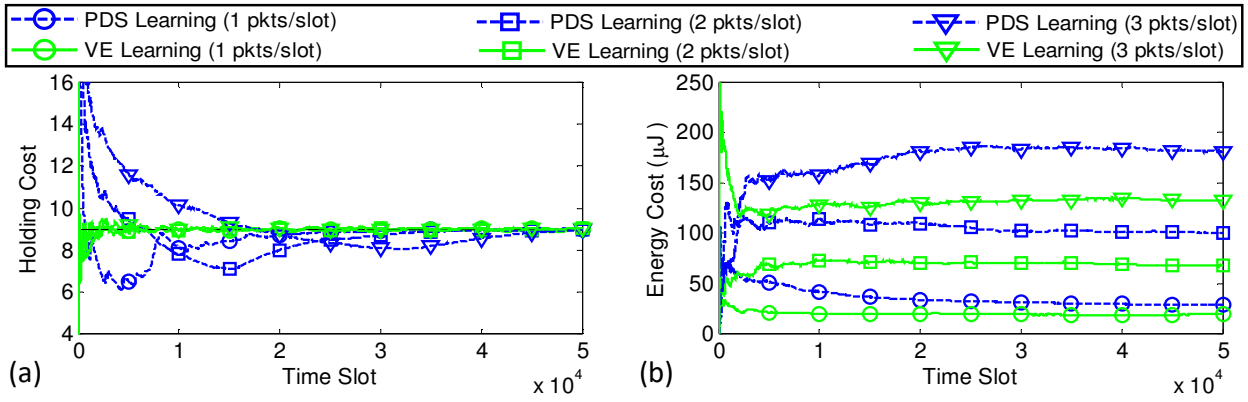


Figure 7: Comparison between the proposed virtual experience learning algorithm and PDS learning using the proposed rate-adaptive CSMA/CA protocol when users have heterogeneous arrival rates. Users 1, 2, and 3 have arrival rates 1, 2, and 3 packets/slot, respectively. All users have the same holding cost constraint (9 packets). (a) Cumulative average holding cost vs. time. (b) Cumulative average energy cost vs. time.

Approved for Public Release; Distribution Unlimited.

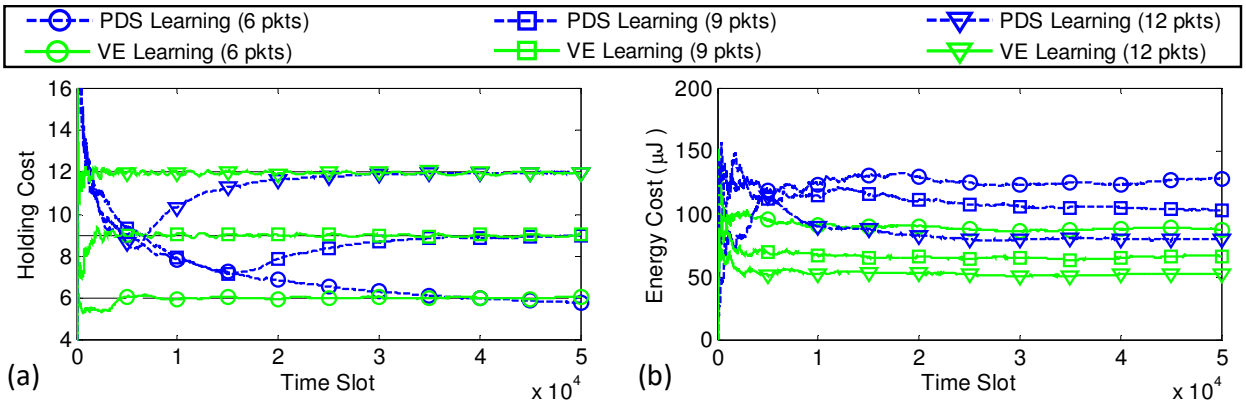


Figure 8: Comparison between the proposed virtual experience learning algorithm and PDS learning using the proposed rate-adaptive CSMA/CA protocol when users have heterogeneous holding cost contraints. Users 1, 2, and 3 have holding cost constraints 6, 9, and 12, respectively. All users have the same arrival rate (2 packets/slot). (a) Cumulative average holding cost vs. time. (b) Cumulative average energy cost vs. time.

4.3.1 Impact of Arrival Rates and Holding Cost Constraints

In Fig. 7, we compare the cumulative average holding and energy costs achieved by three users with heterogeneous arrival rates (and fixed holding cost constraints) when they deploy virtual experience learning and PDS learning. In both scenarios, we use the proposed rate-adaptive CSMA/CA protocol. It is clear that virtual experience learning converges dramatically faster than PDS learning. Indeed, the holding (energy) cost converges in approximately 5K (10K) slots under the virtual experience learning algorithm, while taking up to 50K (far beyond 50K) slots under PDS learning. Furthermore, it is clear that it takes longer to converge to the optimal solution (particularly for PDS learning) when there are higher arrival rates (which, for a fixed holding cost constraint, correspond to tighter delay constraints). Higher arrival rates also require the expenditure of more energy. Fig. 8 shows similar results as Fig. 7, but in a scenario where the three users have heterogeneous holding cost constraints (and fixed arrival rates). Again, we observe that virtual experience learning converges faster than PDS learning, and that meeting tighter constraints requires longer convergence times and more energy. Across the results in Fig. 7 and Fig. 8, virtual experience learning reduces energy consumption by 27%-35% compared to PDS learning.

4.3.2 Comparison to Conventional CSMA/CA

In Fig. 9, we compare the average energy and delay obtained by 10 users over ten 10,000 time slot simulations when they operate under the proposed CSMA/CA protocol and the conventional CSMA/CA protocol (i.e., IEEE 802.11 DCF). In both scenarios, we use the virtual experience learning algorithm to optimize the users' scheduling policies. We observe that the proposed solution enables all users to meet their holding cost constraints and consumes 65% less energy across users than the conventional CSMA/CA protocol. Although conventional CSMA/CA can approximately meet loose constraints, it is clear that it cannot meet tighter constraints. Indeed, 5 users are unable to meet their holding cost constraints,

Approved for Public Release; Distribution Unlimited.

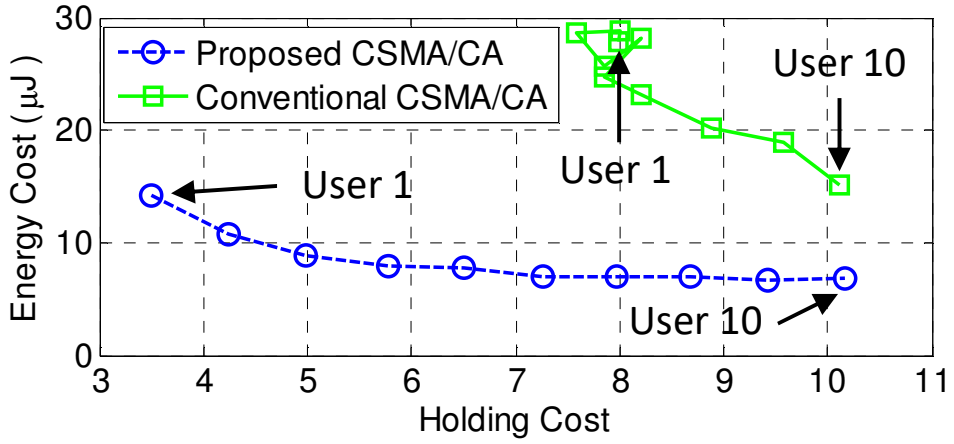


Figure 9: Comparison between the proposed CSMA/CA protocol and the conventional CSMA/CA protocol (i.e., the IEEE 802.11 DCF) using virtual experience learning. Each point corresponds to one user's average energy cost and average holding cost averaged over ten 10,000 time slot simulations. The lines are included for reference and should not be interpreted as tradeoff curves. 10 users are simulated with holding cost constraints ranging from 3.5 to 10.25 packets. All users have the same arrival rate (0.5 packets/slot).

which are below 8 packets. This happens because the conventional CSMA/CA protocol gives channel access to users that select the highest rates less than 45% of the time, whereas the proposed solution gives channel access to these users more than 70% of the time.

4.3.3 Discussion

As part of this effort, we investigated learning-based energy-efficient multi-user scheduling of delay-sensitive data over fading channels. We showed that the multi-user problem is intractable and approximately solved it by decomposing it into multiple (coupled) single-user rate adaptation problems. We proposed a reinforcement learning algorithm to solve the single-user problems online, thereby enabling users to minimize their energy consumption subject to their delay constraints, even though the channel, traffic arrival, and multi-user dynamics are unknown a priori. We designed a rate-adaptive CSMA/CA protocol that works in tandem with the rate adaptation algorithm at the physical layer to enable users to consume less energy than the conventional 802.11 DCF, while allowing them to meet their delay constraints, which, in general, the 802.11 DCF cannot do. In addition, the proposed learning algorithm converges significantly faster than a state-of-the-art solution.

Approved for Public Release; Distribution Unlimited.

5. *UB-ANC Drone: A Flexible Airborne Networking and Communications Testbed*

5.1 Introduction

Thus far, we have addressed the first objective of this project, which was to establish new priority-and deadline driven scheduling solutions. In the next three sections, we address the second objective of this project, which is to develop an experimental and simulation-based framework for evaluating airborne networking and communications protocols in the context of the overlaying application/mission. We do this by leveraging small form factor unmanned aerial vehicles (UAVs).

Networked UAVs have emerged as an important technology for public safety, commercial, and military applications including surveillance [49–51], search-and-rescue [11, 52–55], emergency first response [56], package delivery [57–60], environmental monitoring [61–63], and precision agriculture [64–71]. However, designing, implementing, and testing UAV networks poses numerous interdisciplinary challenges because the communications and networking problems cannot be explored independently of aero-mechanical, sensing, control, embedded systems, and robotics challenges. Indeed, UAV networks are fundamentally *cyber-physical systems* [4].

Although the physical characteristics of a UAV network can be simulated [1, 4, 72, 73], actual implementations and field-tests have been recognized as crucial for demonstrating and evaluating solutions in real-world operating environments [74–76]. Unfortunately, there is currently no suitable experimental testbed framework enabling researchers to holistically explore these challenges. To address this problem, in this project, we developed a software-defined UAV networking platform at the University at Buffalo (UB). The platform, which we call UB's Airborne Networking and Communications Testbed (UB-ANC), combines quadcopters that are capable of autonomous flight with sophisticated command and control capabilities and software-defined radios (SDRs[1]), which enable flexible deployment of novel communications and networking protocols. In particular, UB-ANC provides us the ability to collect data to measure and understand the connection between the underlying networking and communications capabilities and the ability of the UAVs to effectively accomplish different tasks in different network environments.

In this section, we describe the design and implementation of UB-ANC. Our contributions

[1]Note that UB-ANC's software architecture also accommodates off-the-shelf wireless networking technologies, e.g., Wi-Fi, Zigbee, and LTE.

Approved for Public Release; Distribution Unlimited.

are as follows:

1. We define a modular and extensible open platform with reconfigurable communications and networking capabilities, which can be easily modified for rapidly testing novel protocols at different layers of the protocol stack. UB-ANC not only supports off-the-shelf wireless networking technologies (e.g., Wi-Fi, Zigbee, LTE), but also supports custom software-defined wireless technologies. To the best of our knowledge, UB-ANC is the first aerial networking testbed that leverages SDR transceivers.

2. We leverage source code from a popular open-source ground station (APM Planner 2) to enable sophisticated command and control capabilities among drones. Unlike conventional setups, where a remote laptop is used as a ground station to monitor and control a drone over a telemetry link, we equip every drone with an on-board embedded computer that runs a simplified version of the ground station software.

3. The on-board ground station is built around the popular open-source Micro Air Vehicle Communications Protocol (MAVLink), which specifies message formats for communication between ground stations and MAVLink compatible flight controllers. The on-board ground station can send commands to the on-board flight controller or to other drones in the network. In this way, our platform supports both centralized and distributed mission planning, and allows missions to be planned statically or dynamically.

4. Our proposed framework works with all MAVLink compatible flight controllers. Consequently, UB-ANC not only supports different flight controllers, but also many different types of vehicles including rovers, boats, planes, helicopters, and multirotors.

The rest of the section is organized as follows. In Section 5.2, we discuss related work. In Sections 5.3.1 and 5.3.2, we introduce UB-ANC's hardware and software architectures, respectively. We conclude in Section 5.4.

5.2 Related Work

There has been a lot of interest in the research community on UAV networking. In [73], Rohrer et al. develop a domain-specific architecture and protocol suite for cross-layer optimization of airborne networks. They introduce a TCP-friendly transport protocol, IP-compatible network layer, and geolocation aware routing. They perform simulations of their protocols in network simulator programs (ns-2 and ns-3). In [1], the authors propose the Mobility Aware Routing / Mobility Dissemination Protocol (MARP/MDP) to reduce latency and routing overheads by exploiting the known trajectories of airborne nodes. The nodes' trajectories are preplanned to maximize network connectivity using techniques in [77]. MARP/MDP is compared against OLSR and AODV using the QualNet network simulator. In [72], Le et al. simulate a reliable user datagram protocol in OPNET Modeler v14.5 and in [4], Namuduri et al. discuss cyber-physical aspects of airborne networks and use ns-2 to study average path durations under different node velocities, hop counts, and node densities. While this prior work contributes significantly to the advancement of UAV networking

Approved for Public Release; Distribution Unlimited.

protocols and understanding some cyber-physical aspects of UAV networks, the protocols have not been implemented and tested in a real system.

In [74], Allred et al. study airborne wireless sensor networks for atmospheric, wildlife, and ecological monitoring. They equip airborne nodes with off-the-shelf 802.15.4-compliant Zigbee radios. They perform experiments to evaluate the performance of air-to-air, air-to-ground, and ground-to-ground wireless links, as well as network connectivity. In [76], researchers at the University of Colorado test the performance of off-the-shelf IEEE 802.11b (Wi-Fi) networking equipment in an airborne mesh network. They show that a mesh network can extend the communication range among airborne nodes in a small unmanned aerial system (UAS), they explore how a mesh network can be used to enable a remote operator to send command and control information to distant aircraft and to receive telemetry information back over the network, and they use controlled mobility to enable ferrying of delay-tolerant data between nodes in a fractured/partitioned network.

Recently, researchers have started investigating the benefits of equipping drones with SDRs [78–80]. In [78], dos Santos et al. design and implement a drone equipped with a low-cost SDR receiver, which can automatically track wildlife tagged with very high frequency (VHF) radio collars. In [79], Jakubiak implements a drone equipped with a low-cost SDR receiver to gather data about the coverage of a cellular network. In [80], Zhou envisions a system for railways in which drones relay data for passengers in high-speed trains to different networks (e.g., satellite or cellular). While [78, 79] implement systems with only SDR receivers, [80] does not provide any system implementation.

In summary, while a lot of significant contributions have been made in designing and implementing UAV networks, which exploit communications and networking technologies for command and control, telemetry, and coordination among multiple agents, existing system implementations rely on inflexible off-the-shelf transceivers or SDR receivers. In contrast, UB-ANC provides a flexible and highly reconfigurable airborne networking and communications platform for designing, implementing, and testing state-of-the-art communications and networking protocols in conjunction with sophisticated mission planning algorithms. While the proposed framework is designed to be compatible with off-the-shelf wireless interfaces, e.g., Wi-Fi, Zigbee, and LTE, to the best of our knowledge, UB-ANC is the first UAV networking platform designed to support SDR transceivers. In general, this provides researchers more flexibility to design, implement, and test new communications and networking protocols for UAVs.

5.3 Methods, Assumptions and Procedures

5.3.1 Hardware Components

In this section, we describe the high-level hardware architecture of a UB-ANC Drone. We introduce the core components of a drone that are required to use the UB-ANC platform, while also showing that UB-ANC is flexible and can work in numerous configurations.

There are three main hardware components on-board a UB-ANC Drone: a flight controller, an embedded computer, and a wireless network element. Table 4 shows two unique

Approved for Public Release; Distribution Unlimited.

Table 4: Comparison between two UB-ANC drone configurations.

	SDR Configuration	Wi-Fi Configuration
Flight Controller	Pixhawk	Pixhawk
Embedded Computer	USRP E310 / Dual Core ARM Cortex-A9	Raspberry Pi 2 / Quad-Core ARM Cortex-A7
Wireless Technology	USRP E310 SDR	Wi-Fi

drone configurations, although many others are possible. Both configurations use a Pixhawk[2] flight controller; however, as we will see in Section 5.3.2, UB-ANC's software architecture is compatible with many other popular flight controllers. Note that, in both configurations, the Pixhawk is connected to the embedded computer through a USB interface.

The differences between our two drone configurations arise from the choice of wireless network technology. The first configuration uses a USRP E310 SDR[3] from Ettus Research for communication; however, other embedded SDRs could be used instead (e.g., the USRP B200-mini[4] or the bladeRF[5]). The USRP E310 includes a 667 MHz dual-core ARM Cortex-A9 processor; therefore, the USRP E310 also servers as the embedded computer. This configuration is designed for developing new communications and networking protocols for UAVs.

The second configuration uses a Wi-Fi module for communication and a Raspberry Pi 2 as the embedded computer; however, other wireless network technologies (e.g., Zigbee or LTE) and other embedded computers (e.g., Beagleboard[6] or ODROID[7]) could be used instead. This configuration is best suited for applied UAV networking research where the focus is not on the specific communications and networking protocols, but on using multiple networked UAVs to accomplish a task.

Figure 10 shows one of our three custom-built UB-ANC drones in the SDR configuration. It achieves over 25 minutes of flight time while carrying the 400 g USRP E310 as its payload (for a total weight of 3.125 kgs).

5.3.2 Software Components

Now that we know the hardware requirements of a UB-ANC Drone, we are ready to describe UB-ANC's software architecture. Recall from Section 5.3.1 that a UB-ANC Drone includes a flight controller and an embedded computer. In our setup, the embedded computer runs Yocto Linux[8] as its operating system and the flight controller runs ArduPilot APM:Copter[9] as its firmware. The systems are connected to each other using USB CDC-ACM as a serial port with baud rate 115200 bps.

Figure 11(a) provides a high-level diagram of UB-ANC's core software architecture, which

[2] http://copter.ardupilot.com/wiki/common-pixhawk-overview/
[3] https://www.ettus.com/product/details/E310-KIT
[4] https://www.ettus.com/product/details/USRP-B200mini-i
[5] http://www.nuand.com/
[6] https://beagleboard.org/
[7] http://magazine.odroid.com/odroid-xu4/
[8] https://www.yoctoproject.org
[9] http://copter.ardupilot.com

Approved for Public Release; Distribution Unlimited.

Figure 10: A UB-ANC drone (SDR configuration).

comprises four components: the Agent Control Unit (ACU), the Network Control Unit (NCU), the MAVLink Control Unit (MCU), and the Logging Unit (LU). The ACU is the "brains" of a UB-ANC drone: it contains the mission planning logic and interfaces with (i) the NCU to talk with different network elements; (ii) the MCU to talk with different flight controllers; and (iii) the LU to log status information. Table 5 provides details about the APIs that the ACU uses to interface with the NCU and the MCU. Note that the list of methods in Table 5 is illustrative, but not exhaustive.

The aforementioned software components are implemented using Qt[10], which is an object-oriented C++ cross-platform application development framework. We have chosen Qt as the main application framework based on the following considerations:

- It facilitates event-driven programming and makes it easy to maintain a modular design. Specifically, using Qt's signals and slots mechanism, components can communicate by emitting signals and capturing other components' signals using slots.

- It is a stable open-source application framework that has been used in many other open-source projects. In particular, some of the open-source software that we are reusing in this project is already implemented using Qt.

[10] http://www.qt.io

Approved for Public Release; Distribution Unlimited.

Table 5: Abbreviated front-end APIs for the Network and MAVLink Control Units (i.e., the NCU and MCU).

Component	Class	Method	Description
Network Control Unit	UBNetwork	getData()	Return data from the receive buffer
		sendData()	Send data to the send buffer
		dataReady()	A signal emitted when data is in the receive buffer
	UBPacket	setSrcID()/getSrcID()	Set/get the source MAV ID for the packet
		setDesID()/getDesID()	Set/get the destination MAV ID for the packet
		setPayload() getPayload()	Set/get the payload for the packet
		packetize() depacketize()	Make/parse the packet stream
MAVLink Control Unit	UASManager	UASCreated()	A signal emitted when a new flight controller is detected
	LinkManager	getLink()	Return the ID of the specific link
		getLinkType()	Return the type of the link (Serial, TCP, ...)
		connectLink()	Connect to the specific link
	UASInterface	setMode()	Set the mode of the flight controller
		getAltitude()	Return the quad-rotor's altitude
		setHeartbeatEnabled()	Enable HEARTBEAT message to the flight controller
		executeCommand()	Send a specific MAVLink command to the flight controller
		isArmed()	Returns 1 if the flight controller is armed; 0 otherwise.
	LinkInterface	setPortName()	Specify the serial port
		setBaudRate()	Set the baud rate of the serial port

- It is a C++ object-oriented framework, which facilitates efficient coding while maintaining high-performance operation.
- It is cross-platform, which makes it easy to port the project across different operating systems, like Windows CE, Custom Embedded Linux, Android, and iOS.

Before we describe each software component in detail, we highlight the key features of the software architecture design:

- **Modularity:** UB-ANC's software architecture is designed to be modular. Each component has a well-defined task so that it can be easily modified and debugged.
- **Extensibility:** The components have well-defined interfaces allowing for easy extensibility. For instance, the NCU and MCU have well-defined front-end and back-end interfaces that allow them to work with different network technologies and different flight controllers, respectively.
- **Utilizing popular open-source standards:** As noted in the introduction of this section, UB-ANC leverages the popular MAVLink protocol; therefore,it supports all

Approved for Public Release; Distribution Unlimited.

MAVLink compatible vehicle controllers including APM[11], Pixhawk[12], Emlid's NAVIO[13], and Intel's Aero[14]. Moreover, since many vehicle controllers that are designed for rovers, boats, planes, helicopters, and multirotors are based on MAVLink, the UB-ANC platform can be easily deployed on different types of vehicles.

In the following subsections, we describe each software component in detail.

5.3.2.1 Agent Control Unit (ACU)

The ACU is responsible for any mission that the drone is supposed to complete. It includes the internal logic for deciding what commands to send to the flight controller (through the MCU) and what information to send to other nodes (through the NCU) to accomplish its mission. In general, the mission planning logic can make decisions based on local state information and information received from other nodes.

The following code shows a finite state machine algorithm for a simple mission where a drone takes off, loiters (i.e., hovers in position), sends a message to another drone, and then lands. The ACU continuously checks the state of the drone and the mission through a function called `missionTracker`, which is called every 10 milliseconds (100 Hz). Each time the `missionTracker` function is called, the ACU checks if the flight controller is armed and then it executes the appropriate function based on the current state of the mission, i.e., `stageStart()`, `stageLoiter()`, or `stageStop()`. A portion of the `stageLoiter()` method is also given below, where `executeCommand()` is used to tell the flight controller to land after the loiter time exceeds a threshold. When the drone finishes loitering, it sends a message to another drone instructing it to start its own simple mission (i.e., takeoff, loiter, and land).

```
void UBAgent::missionTracker() {
    if (!m_uav->isArmed()) {
        return;
    }
    switch (m_stage) {
    case STAGE_START:
        stageStart();
        break;
    case STAGE_LOITER:
        stageLoiter();
        break;
    case STAGE_STOP:
        stageStop();
        break;
    }
}

void UBAgent::stageLoiter() {
```

[11] http://ardupilot.org/copter/docs/common-apm25-and-26-overview.html
[12] http://copter.ardupilot.com/wiki/common-pixhawk-overview/
[13] http://copter.ardupilot.com/wiki/common-navio-overview/
[14] https://software.intel.com/en-us/aero/compute-board

Approved for Public Release; Distribution Unlimited.

```
  if ((QGC::groundTimeSeconds() -
        m_loiter_timer > LOITER_TIME)) {
    m_uav->executeCommand(MAV_CMD_NAV_LAND,
      1, 0, 0, 0, 0, 0, 0, 0, 0);
    m_net->sendData(&m_msg);
    m_stage = STAGE_STOP;
    return;
  }
  ...
}
```

5.3.2.2 Network Control Unit (NCU)

As mentioned earlier, the ACU uses the NCU to send/receive data over the network. For example, one drone can send commands to another drone to visit specific GPS waypoints or, for more sophisticated applications, drones can exchange local state information that their ACUs can use for centralized or distributed mission planning. The NCU is designed so that the underlying network technology can be easily changed while keeping the rest of the system the same. Therefore, we can easily test different wireless network technologies with the same ACU logic so that we can fairly compare the system performance across different configurations.

The NCU provides a front-end API that the ACU uses to access the network. This API comprises the `UBNetwork` and `UBPacket` classes as shown in Table 5. The NCU's back-end uses an interprocess communication (IPC) mechanism (a local socket) with a well-defined packet format (Source MAV ID, Destination MAV ID, Payload) to connect to the wireless network. Thus, the NCU can be viewed as the application layer in the network protocol stack.

The NCU's back-end interface is shown in Figure 11a as a bi-directional arrow labeled "Local socket to/from network." While the NCU and its front / back-end interfaces are well-defined, everything beyond the back-end depends on the underlying network technology (e.g., Wi-Fi, Zigbee, LTE, or a software-defined technology). For example, in Figure 11b, we show how the NCU interfaces with an SDR where the transport, network, data link/MAC and physical layers are implemented within GNU Radio [81]. As another example, in Figure 11c, we show how the NCU interfaces with a local proxy, which uses the existing networking infrastructure of the operating system to connect to a standard wireless network (e.g., Wi-Fi, Zigbee, or LTE). In both Figures 11b and 11c, the connection to the NCU is shown as a bi-directional arrow labeled "Local socket to/from NCU." Note that, while the back-end of the NCU that connects to the local proxy is well-defined, the interface from the local proxy to the wireless network is specific to the underlying wireless network technology.

The ACU uses the NCU to send/receive data over the network as follows. When the ACU sends a packet to the NCU, the NCU puts the packet into a private queue called `m_send_buffer` and then sends the packet to the wireless network using the aforementioned IPC mechanism. When a packet is received by the NCU from the network, it raises a signal (`dataReady()`) to notify the ACU that there is a packet in the `m_receive_buffer` buffer. The ACU then reads the buffer and processes the received packet. The following code shows

Approved for Public Release; Distribution Unlimited.

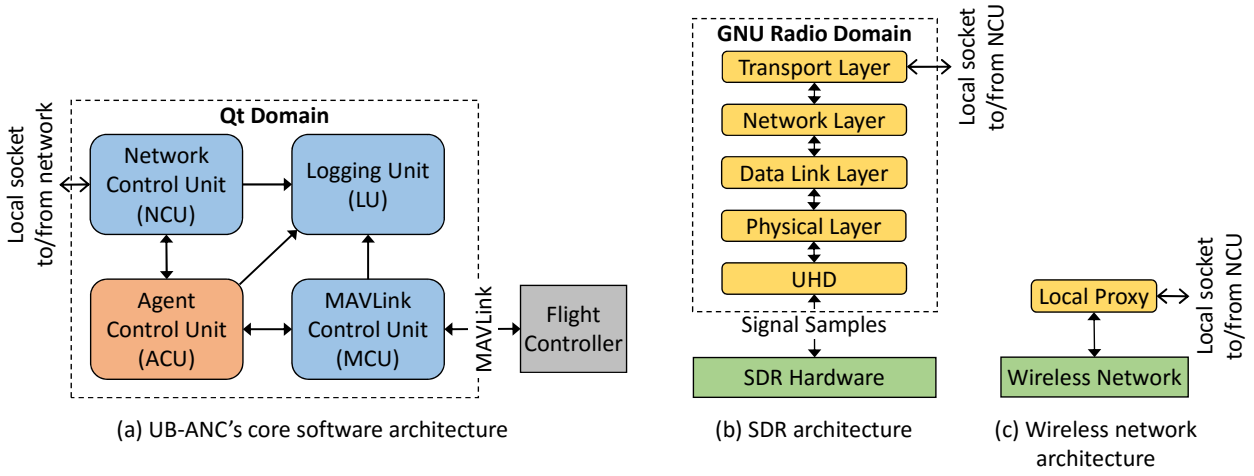


Figure 11: High-level software architecture diagram. (a) UB-ANC's core software architecture with its interface to the network. (b) SDR architecture with its interface to the Network Control Unit. (c) Standard wireless network architecture with its interface to the Network Control Unit.

how the `sendData()` and `getData()` methods are implemented in the `UBNetwork` class using a Qt container to buffer and unbuffer the data. Notice that, before a packet is queued at the sender, it is first packetized using methods from the `UBPacket()` class.

```
void UBNetwork::sendData(quint8 desID,
       const QByteArray& data) {
    UBPacket packet;
    packet.setSrcID(m_id);
    packet.setDesID(desID);
    packet.setPayload(data);
    QByteArray* stream =
      new QByteArray(packet.packetize());
    m_send_buffer.enqueue(stream);
    ...
}

QByteArray UBNetwork::getData() {
    QByteArray data;
    if (m_receive_buffer.isEmpty())
        return data;
    QByteArray* stream = m_receive_buffer.dequeue();
    data = *stream;
    delete stream;
    return data;
}
```


Approved for Public Release; Distribution Unlimited.

Table 6: An abbreviated list of MAVLink commands.

CMD ID	Command Name	Description
16	MAV_CMD_NAV_WAYPOINT	Navigate to a waypoint
19	MAV_CMD_NAV_LOITER_TIME	Loiter around a waypoint for X seconds
20	MAV_CMD_NAV_RETURN_TO_LAUNCH	Return to launch location
21	MAV_CMD_NAV_LAND	Land at location
22	MAV_CMD_NAV_TAKEOFF	Takeoff from ground
176	MAV_CMD_DO_SET_MODE	Set system mode
183	MAV_CMD_DO_SET_SERVO	Set a servo to a desired PWM value

5.3.2.3 MAVLink Control Unit (MCU)

The MCU provides a front-end API that the ACU uses to send commands to (and receive messages from) a flight controller. The back-end of the MCU supports different types of connections to the flight controller (e.g., USB, Ethernet, and serial) and can even connect to multiple flight controllers simultaneously.[15] The MCU communicates with the flight controller using the MAVLink messaging protocol[16]; consequently, the MCU can easily interface with any MAVLink compatible flight controller.

MAVLink supports various messages and commands.[17] One of the most important messages, called the HEARTBEAT, is generated by the MCU and flight controller every second (1 Hz). The HEARTBEAT message shows that the link between the MCU and flight controller is still alive. If the HEARTBEAT message from the MCU is lost, then the flight controller goes into a preconfigured failsafe mode (either return-to-launch, which requires a GPS lock, or land, which does not). On the other hand, the HEARTBEAT message from the flight controller contains information that the MCU can use for different tasks. This information includes, but is not limited to, the type of micro air vehicle (quadcopter, helicopter, fixed wing, etc.); the type of flight controller (APM, Pixhawk, etc.); the mode of the flight controller (armed, autonomous, manual, stabilize, etc.); and the MAVLink protocol version. Note that not all MAVLink commands are supported by all flight controllers. Therefore, knowledge of the specific type of flight controller is important to ensure that only the correct commands are used.

Table 6 shows an abbreviated list of some important MAVLink commands. Every MAVLink command is associated with up to seven parameters. For illustration, the parameters of the loiter command (`MAV_CMD_NAV_LOITER_TIME`), which include the loiter duration, latitude, longitude, and altitude, are shown in Table 7. The ACU uses the `executeCommand()` method to send specific MAVLink commands to the flight controller (see Table 5). A code snippet in Section 5.3.2.1 shows how to use the `executeCommand()` method.

The MCU is implemented using four classes from an open-source project called APM Planner 2, namely, `UASManager`, `LinkManager`, `UASInterface`, and `LinkInterface`.[18] APM

[15] In general, it is possible for a vehicle to have multiple controllers. For example, a vehicle that can switch between air, land, and water may have a separate controller for each modality.

[16] http://qgroundcontrol.org/mavlink/start

[17] https://pixhawk.ethz.ch/mavlink

[18] https://github.com/diydrones/apm_planner

Approved for Public Release; Distribution Unlimited.

Table 7: Parameters for the loiter command.

Param No.	Description
1	Seconds (decimal)
2	Empty
3	Radius around the waypoint, in meters
4	Desired yaw angle
5	Latitude
6	Longitude
7	Altitude

Planner 2 is a GUI-based ground station that can be used to define missions, send missions to a flight controller, and track a drone on a map. It is based on Qt and works with MAVLink compatible flight controllers. As we noted in the introduction of this section, GUI-based ground stations like APM planner 2 are typically loaded on a laptop to monitor and control a drone over a telemetry link; however, in order to support more sophisticated mission planning algorithms than conventional setups (which rely on centralized control), we load ground station software directly onto each drone's embedded computer (enabling fully distributed control). To achieve this, we carefully stripped away the GUI-based elements of the aforementioned classes to create a light-weight console-based ground station.

LinkManager and LinkInterface: The `LinkManager` class is responsible for managing different kinds of links between the flight controller and the MCU (serial link, TCP/UDP link, telemetry link, etc.). Every link has a corresponding class (`SerialLinkInterface`, `TCPLink`, `UDPLink`, etc., which are all derived from the base link class `LinkInterface`). When a link is established, the `LinkManager` creates the corresponding link object. The ACU then uses the link object to control the link (connect, disconnect, set baud rate, etc.).

UASManager and UASInterface: The `UASManager` class is responsible for managing different kinds of flight controllers (APM, Pixhawk, etc.). When the MCU receives a HEARTBEAT message, the `UASManager` first determines the type (and ID) of the flight controller that sent the message. If the corresponding flight controller's object does not already exist, then the `UASManager` creates the appropriate flight controller object (`ArduPilotMegaMAV`, `PxQuadMAV`, etc., which are all derived from the base flight controller class `UASInterface`) and puts it in a private list called `m_uas_list`. The ACU then uses the flight controller object to send commands to (and receive messages from) the corresponding flight controller.

5.3.2.4 Logging Unit (LU)

There is a lot of information that can be tracked in the system including, but not limited to, GPS position (longitude, latitude, and altitude), MAVLink messages, drone ground speed, packet information (e.g., packet ID, source ID, and destination ID), channel state information, etc. We track data in our system using QsLog[19], which is a system logger based on Qt's `QDebug` class. The data can be logged on a MicroSD card so it can be analyzed offline, or it can be sent to a ground station where it can be viewed and analyzed in real-time. The Logging Unit can be configured to provide different levels of verbosity using different logging

[19] https://github.com/victronenergy/QsLog

Approved for Public Release; Distribution Unlimited.

Table 8: Abbreviated mission log.

Time Stamp	Event Description
2016-02-08T16:19:57.637	Mode changed to Stabilize
2016-02-08T16:19:57.639	Calibrating barometer
2016-02-08T16:20:46.188	Arming motors
2016-02-08T16:20:54.813	Mode changed to Loiter
2016-02-08T16:21:19.406	Mode changed to Land
2016-02-08T16:21:31.773	Mission complete

functions, e.g., `QLOG_ERROR()`, `QLOG_WARN()`, and `QLOG_DEBUG()`.

In Table 8, we show an abbreviate log for the simple takeoff, loiter, and landing mission described in Section 5.3.2.1, which we tested on one of our UB-ANC drones. The log shows time stamps for key events (with millisecond granularity) along with the corresponding event descriptions. In Table 8, we see that the flight controller is initialized to the "Stabilize" mode (a simple manual flight mode) and then its barometer is calibrated. After some delay, the motors are manually armed using an RC remote, which triggers the autonomous mission to start. Once armed, the quadcopter takes off and climbs in altitude. After approximately 9 seconds, it switches to "Loiter" mode and hovers for approximately 25 seconds before it switches to "Land" mode. The mission ends when the drone lands.

5.4 Results and Discussion

We introduced the hardware and software architecture of the University at Buffalo's Airborne Networking and Communications Testbed (UB-ANC). To the best of our knowledge, UB-ANC is the first aerial networking platform that combines quadcopters capable of autonomous flight with sophisticated mission planning capabilities and flexible SDR-based transceivers, while also supporting off-the-shelf transceivers like Wi-Fi, Zigbee, and LTE. UB-ANC is designed to be modular and extensible in terms of both hardware and software, and it is built around popular open-source software and standards to facilitate its adoption. Although we present UB-ANC in the context of quadcopters, it can be used for other types of multirotors as well as helicopters, planes, boats, and rovers. UB-ANC is an open-source project available via GitHub:

https://github.com/jmodares/UB-ANC

Approved for Public Release; Distribution Unlimited.

6. *UB-ANC Emulator: An Emulation Framework for Multi-Agent Drone Networks*

6.1 Introduction

In Section 5, we introduced the UB-ANC Drone platform and the UB-ANC Agent software, which facilitate the design and deployment of multi-drone networks and applications. However, there are numerous challenges associated with conducting field tests with networked MAVs. On the technical side, the systems and software on each MAV including the network protocol stack, the mission planning algorithms, and the flight-controller are incredibly complex. While each component can be tested independently, testing the fully integrated system is non-trivial. Listed below are some challenges from our experience:

- Conducting field tests requires having multiple flight-ready MAVs. This is challenging because MAVs require frequent and time consuming maintenance, especially when experimenting with large numbers.
- Conducting field tests requires good weather conditions (no rain, low wind, etc.).
- Conducting field tests requires FAA Part 107 certified remote pilots.
- MAVs have limited battery lifetimes ($<$ 30 minutes) mandating a large supply of batteries and frequent charging interruptions during experimentation.

A popular approach to ease deployment challenges is the use of simulation/emulation. There are several simulators that address parts of the challenges listed above [82,83]. Robotics simulation packages allow for simulation of individual MAVs with realistic physics, but make it hard to simulate/emulate multiple drones at the same time and do not provide network modeling at a reasonable fidelity. Network simulators support the simulation of wireless networks, but do not make it easy to simulate the interaction between networking, mission planning, and control. We looked through several possibilities and found it challenging to assemble a set of existing tools that would help us test MAV networking applications in simulation and translate them to practice with ease.

To mitigate these challenges, we developed the *UB-ANC Emulator*: a simulation framework that makes it easy to design, implement, and test various drone networking applications in simulation and transition them to actual drones seamlessly. It has been designed using open-source software components that are borne out of the popular hobby drone movement and is therefore easily usable with several off-the-shelf as well as custom-built drones.

Approved for Public Release; Distribution Unlimited.

The UB-ANC Emulator uses the same software that executes on the actual drone hardware including a software-in-the-loop (SITL) simulator of the flight controller, the protocol for communicating with the flight controller [i.e., the Micro Air Vehicle Communication Protocol (MAVLink [84]) described in Section 5.3.2.3], mission planning algorithms, and the application program interfaces (APIs) for the network and sensors. It also provides the same data logging capabilities as the actual drones and the ability to monitor the emulated mission via real-time visualization using, e.g., APM Planner [85], QGroundStation [86], or MAVProxy [87], which can track, monitor, and log the drones' movements. It is also designed to be both modular and extensible, and can therefore be extended to easily incorporate other network elements, sensors, planning algorithms and flight controllers that use the MAVLink protocol.

6.2 Related Work

Deploying and experimenting with aerial networks requires expertise in several areas including multi-agent systems, robotics, and mobile ad-hoc and wireless networks. A lot of research in each of these areas has been fueled by powerful simulation environments.

Multi-agent systems have been studied for several decades with significant interest in modeling coordination and swarm behavior. Swarm and MASON allow simulation of hundreds of agents and their interaction. Swarm [88] was built in the early 90s and fueled the beginning of swarm research. MASON [89] is written in Java, and distinguishes between modeling and visualization allowing for easy attachment and detachment during runtime, and enabling algorithm developers to easily debug swarm applications. These platforms allow simulation of a large number of agents, but it is not easy to represent an off-the-shelf aerial vehicle in them. Similarly, Simbeeotic [90] was designed to simulate behavior of swarms of MAVs. It simulates full physics using the JBullet physics engine and several multi-UAV applications have been demonstrated on it. However, it does not have the ease-of-transition from simulation to experiment. In fact, it would require detailed customization of the simulator for its use with current hobby hardware/software.

There has been a lot of interest from academia and industry in all aspects of robotics. A lot of this research is powered by strong simulation support. In the 2000s, Player-Stage [91] was one of the first simulators that allowed for development of controllers that could be deployed on robots, as well as in simulation. Stage is a 2.5D simulation engine that provides realistic physics simulation. ROS [92] evolved the server-client architecture used for communication between controller nodes in Player-Stage to a peer-peer distributed architecture. It is distributed with Gazebo, a realistic 3D simulator with full 6-DoF physics simulation. While these systems form excellent simulation/emulation platforms for robotic algorithms, they are challenging to scale up. As the number of robots (and correspondingly the number of nodes) grow in simulation, the time to simulate grows because they simulate full physics. It is near impossible to simulate tens to hundreds of agents interacting in such systems.

Many advances in both wired and wireless networking have been driven by simulation tools. ns-2 [93] and ns-3 [94] are discrete event network simulators that have been used both in the classroom and by researchers for over a decade. Opnet [95] and Glomosim [96] have also been used for wireless networking research. Such simulators provide realistic networking

Approved for Public Release; Distribution Unlimited.

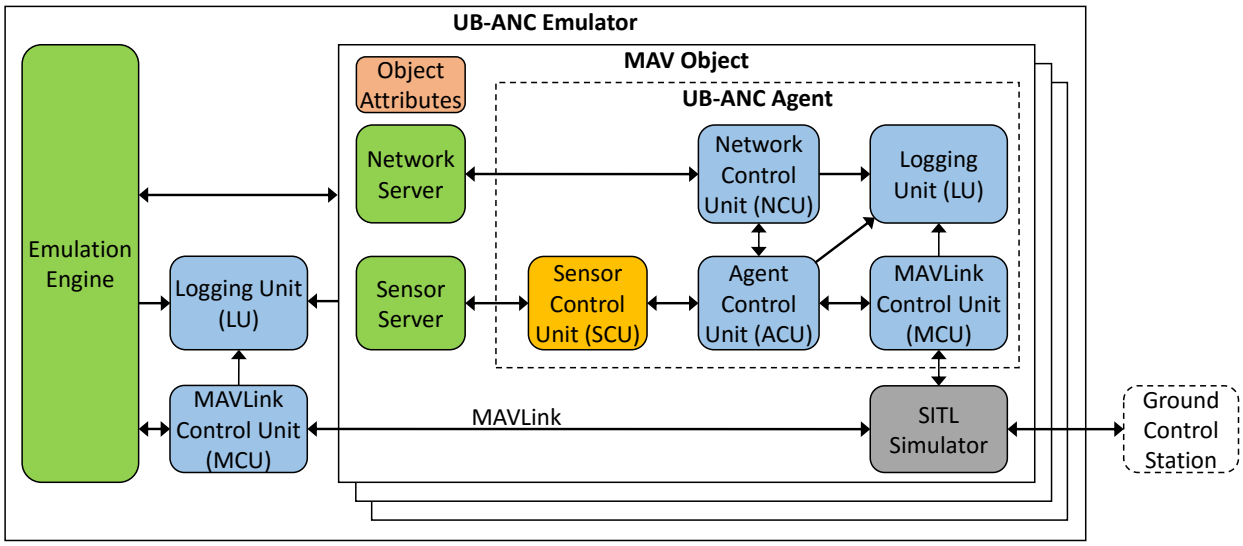


Figure 12: The UB-ANC Emulator's software architecture.

including queuing behaviors, protocol interaction, and channel modeling. They do not, however, model physical movement of individual nodes and related dynamics accurately.

While there are several platforms that address aspects of the development of airborne networks, our survey did not find a platform that was adequately suited for drone networking research. We also believe that enabling simulation/emulation of aerial vehicle platforms that have evolved from popular open-source standards will allow for quick, seamless, and low-cost deployment on real systems. The UB-ANC Emulator represents our effort to bridge the gap between research and deployment on such systems.

6.3 Methods, Assumptions and Procedures

6.3.1 Software Architecture

Fig. 12 provides a high-level diagram of the UB-ANC Emulator's architecture, which comprises three main components: the Emulation Engine, MAV Object, and UB-ANC Agent. The Emulation Engine is the core of the emulator. It coordinates various tasks, and instantiates and interfaces with one MAV Object per simulated MAV. Each MAV Object contains the UB-ANC Agent software, which hosts the mission behavior and can be directly executed on the drone (see Section 5.3.2). Each UB-ANC Agent interfaces with three other modules: a flight controller, a network server, and a sensor server. An open-source software-in-the-loop (SITL) simulator [97] is used to simulate the flight controller. The SITL simulator can be connected to an open-source GUI such as APM Planner [85] to visualize and monitor the emulated MAVs. We now describe each component of the emulator in detail.

Approved for Public Release; Distribution Unlimited.

6.3.1.1 UB-ANC Agent

As described in Section 5.3.2, the UB-ANC Agent comprises of four components: the Agent Control Unit (ACU), the Network Control Unit (NCU), the MAVLink Control Unit (MCU), and the Logging Unit (LU). The ACU is the "brains" of a UB-ANC drone: it contains the mission planning logic and interfaces through well-defined APIs with (i) the NCU to talk with different network elements; (ii) the MCU to talk with different flight controllers; and (iii) the LU to log status information. Importantly, the UB-ANC Agent software can be moved from simulation to experimentation (i.e., actual deployment on drones) by changing only one line of code.

6.3.1.2 MAV Object

The MAV Object component represents a MAV in the emulator. Along with an instance of the UB-ANC Agent, it contains an instance of the SITL simulator [97], which simulates the flight controller that interfaces with the UB-ANC Agent via MAVLink messages. The MAV Object also creates Network Server and Sensor Server components. These provide a level of indirection, and abstract the individual network and sensor elements present in simulation which allows for progressive emulation. For example, we can simulate the drone behavior but connect the computer to a wireless network allowing for network experimentation while simulating the rest of the system (e.g., see Section 6.4.1.3). Similarly, we can simulate any combination of the individual components while testing the rest in experiment.

In simulation, there are several private properties of sensing and communication, such as sensing and communication ranges, and communication and sensing models, which are held in the MAV Object. These are shown as Object Attributes in Fig. 12 and are communicated with the Emulation Engine for reasoning across simulated drones.

6.3.1.3 Emulation Engine

The Emulation Engine is the central housekeeper for the UB-ANC Emulator. It is responsible for instantiating MAV Objects (one for each MAV to be emulated) and sending the mission plans (if any) to them. It also coordinates all the MAV movements, network communication across MAVs, and sensing performed by individual MAVs in simulation. Based on the individual MAV position, by default, it delivers messages to/from MAVs within communication range (as determined by the Object Attributes and the network modality) and sensing information. However, in Section 6.3.2 we develop an API to integrate more sophisticated network simulation capabilities into the Emulation Engine (e.g., ns-3).

6.3.1.4 Software in the Loop (SITL)

Software in the Loop (SITL) [97] simulator is an open source project that simulates the flight dynamics model of a wide variety of vehicle types, including multi-rotor aircraft, fixed-wing aircraft and ground vehicles. Depending on what the simulator is connected to, we can vary the degree of physics being simulated and thereby vary the degree of realism of the simulation. This is a key design choice allowing us to scale up our simulations based on our needs.

Approved for Public Release; Distribution Unlimited.

6.3.2 Network Simulator Integration

By default, the UB-ANC Emulator uses a simple network simulation in which nodes can communicate with each other if they are within a given range. However, this does not accurately reflect the performance of a MAV network where communication links are subject to interference and packet losses, and protocols at the data link, network, and transport layers have a significant influence on network throughput, latency, and reliability. To overcome this limitation, our objective in this section is to allow for realistic network simulation in the UB-ANC Emulator. Our design choices included implementing our own network simulation or integrating an existing network simulator. As discussed in Section 6.2, there has been a lot of research on network simulation. After some exploration, we decided to integrate ns-3, a popular, well-maintained network simulator that has been designed for integration into testbeds and real network stacks.

There are several challenges as well as design choices in integrating a network simulator such as ns-3 into the UB-ANC emulator. We briefly list them below:

- **Clock Synchronization**: ns-3 is a discrete-event simulator that processes events from a queue that are ordered in simulation time. In default operation, simulation time and clock time are not the same because simulation time jumps instantly from one simulated event to the next. However, UB-ANC emulator simulates aerial vehicle behavior in real time.

- **Event Synchronization**: The UB-ANC Emulator simulates networks of MAVs. For this purpose, it translates agent behavior into MAVlink messages that are interpreted by the SITL simulator of the vehicle controller firmware. In short, several MAVlink events get processed in a per-agent manner on the individual SITL simulators. If a network simulator is integrated, it has its own event processing. In order to perform accurate simulation, these two event queues need to be synchronized frequently. This requires the network simulator to be informed of the relevant events in the UB-ANC Emulator in a timely manner.

- **Network Activity Synchronization**: Algorithms implemented in the UB-ANC Emulator will likely perform complex coordination behaviors that rely on network activity. The primary goal of the network simulator integration is to synchronize the network activity between our emulator and the network simulator. This includes exception handling in cases of communication failure and/or external disturbances that affect communication.

This section will discuss how we handle the above challenges, and describe our architecture for realistic network simulation/emulation.

6.3.2.1 API for Network Simulator Integration

Each UB-ANC Agent has an NCU that forwards communication requests from the agent to a network server, which in turn connects to a network element. A network element could be an external network simulator, a wired network interface, a wireless network interface, or any other mechanism used to communicate between MAVs. For the purposes of the current

Approved for Public Release; Distribution Unlimited.

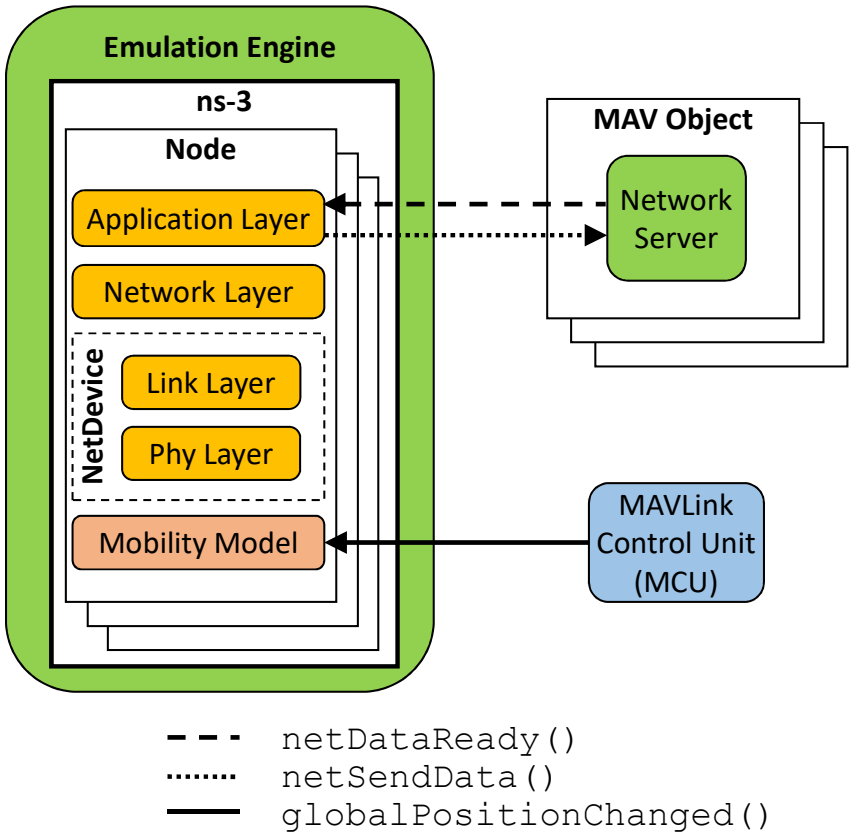


Figure 13: Block diagram illustrating how we have integrated ns-3 into the UB-ANC Emulator.

discussion, we assume that the network element is a network simulator. As mentioned previously, our emulator is built using the open-source Qt framework. We utilize Qt's signals and slots mechanism to communicate between modules.

For *event synchronization*, we forward all relevant events in the UB-ANC Emulator to the network simulator. Given the distributed nature of our implementation, we designed the UB-ANC Emulator to expose the three methods in Table 9 to allow the network simulator to interface with individual MAVs as shown in Fig. 13. These methods not only enable packet transmission and reception, but also track MAV mobility. In this way, a network simulator can realistically model the connectivity and data transmission based on the relative position of the communicating agents.

We now describe how the methods in Table 9 are used for packet transmission, packet reception, and MAV positioning.

Packet Transmission When the ACU wants to send a packet, it forwards the packet to the NCU, which puts it in a private queue called `m_send_buffer`. From there, the packet is forwarded to the sender's corresponding MAV Object using an IPC mechanism and then the MAV Object raises a signal called `netDataReady()`. This signal must be captured by the network simulator so that it can ingest the packet from the MAV Object. Once ingested, the network simulator can process the packet, i.e., send it from the source node to its destination

Approved for Public Release; Distribution Unlimited.

Table 9: API for integrating existing network simulation software into the UB-ANC Emulator.

Method	Description
`netDataReady()`	Signal that is emitted by a transmitter's MAV Object to transfer a packet to a network simulator
`netSendData()`	Slot that is used by a network simulator to transfer a packet to a receiver's MAV Object
`globalPositionChanged()`	Signal that is emitted by the emulator's MCU to inform a network simulator about a drone's updated position

node using an internal representation of the network.

Packet Reception Once a packet is delivered to the destination node in the network simulator, it needs to be sent to the Network Server component of the destination node's MAV Object. This is achieved using the MAV Object's method (slot) `netSendData()`. The Network Server then forwards the packet to the NCU of the corresponding UB-ANC Agent component using an IPC mechanism. Subsequently, the NCU raises a signal called `dataReady()` to notify the ACU that there is a packet in the `m_receive_buffer` buffer. The ACU then reads the buffer and processes the received packet.

Drone Positioning As described earlier, the drones' positions need to be updated in the network simulator to match their positions in the emulator. When a drone's position changes, the emulator's MCU (shown in Fig. 13) raises a signal called `globalPositionChanged()`. The Emulation Engine listens to the signal and passes it along to the network simulator which can then process it accordingly.

6.3.2.2 Ns-3 Integration

A high-level block diagram showing how we integrate ns-3 into the UB-ANC Emulator is provided in Fig. 13. In ns-3, a node represents a mobile transceiver that can send/receive packets to/from other nodes in a simulated network. As shown in Fig. 13, each node contains an application layer, a network layer, a data link layer, a physical layer, and a mobility model. The application layer represents an application that runs on the mobile transceiver and can generate and consume network packets; and the mobility model is responsible for positioning the mobile transceiver in the network over time.

To simulate the MAV network, an ns-3 node (referred to hereafter as node) must be instantiated for each emulated MAV. Each node's application layer handles packet transmit signals (`netDataReady()`) from the corresponding MAV Object in UB-ANC Emulator to initiate a packet transmission. Once a packet is ingested by the source node's application layer, ns-3 sends the packet through its network stack to the destination node. Then, the destination node's application layer uses the received packet slot (`netSendData()`) to send the packet to the node's corresponding MAV Object in our emulator.

As described earlier, a challenge to integrate ns-3 is that it uses a separate scheduler that needs to be synchronized in time with the UB-ANC Emulator's scheduler. For this, we set

Approved for Public Release; Distribution Unlimited.

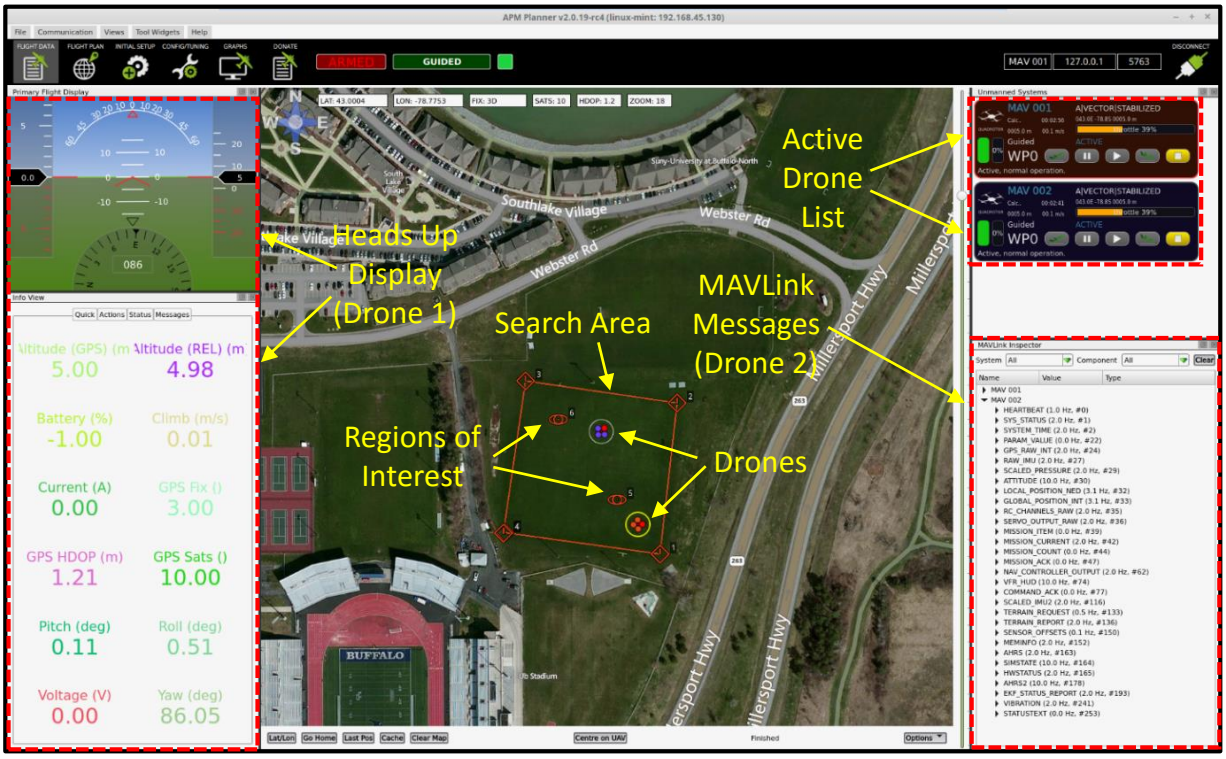


Figure 14: APM Planner visualization for UB-ANC Emulator.

ns-3 to use a real-time scheduler to lock the simulation clock with the CPU clock (and set real time as simulation time). This allows the two schedulers to be synchronized to the same clock.

6.4 Results and Discussion

6.4.1 UB-ANC Emulator Evaluation

In this section, we describe the experiments and simulations that we perform to evaluate the accuracy, scalability, and extensibility of the UB-ANC Emulator (without ns-3). Fig. 10 shows a close up of one of the UB-ANC drones used to gather our experimental results. Fig. 14 shows the emulator connected to APM Planner for visualization.

A major challenge in analyzing a simulator is determining its accuracy with respect to reality. To this end, many robot simulators simulate full physics. However, this makes these simulations not scalable in the number of robots simulated, especially to simulate drone networks. Our simulator simulates MAVlink-compatible robots. The MAVlink protocol communicates in terms of events and their passing. As an indicator of accuracy, we decided to measure the time between events on a UAV and compare it to the corresponding simulator. These results are presented in the next sub-section. We then simulate increasing numbers of UAVs and measure CPU and memory utilization to study scalability of our simulator in the

Approved for Public Release; Distribution Unlimited.

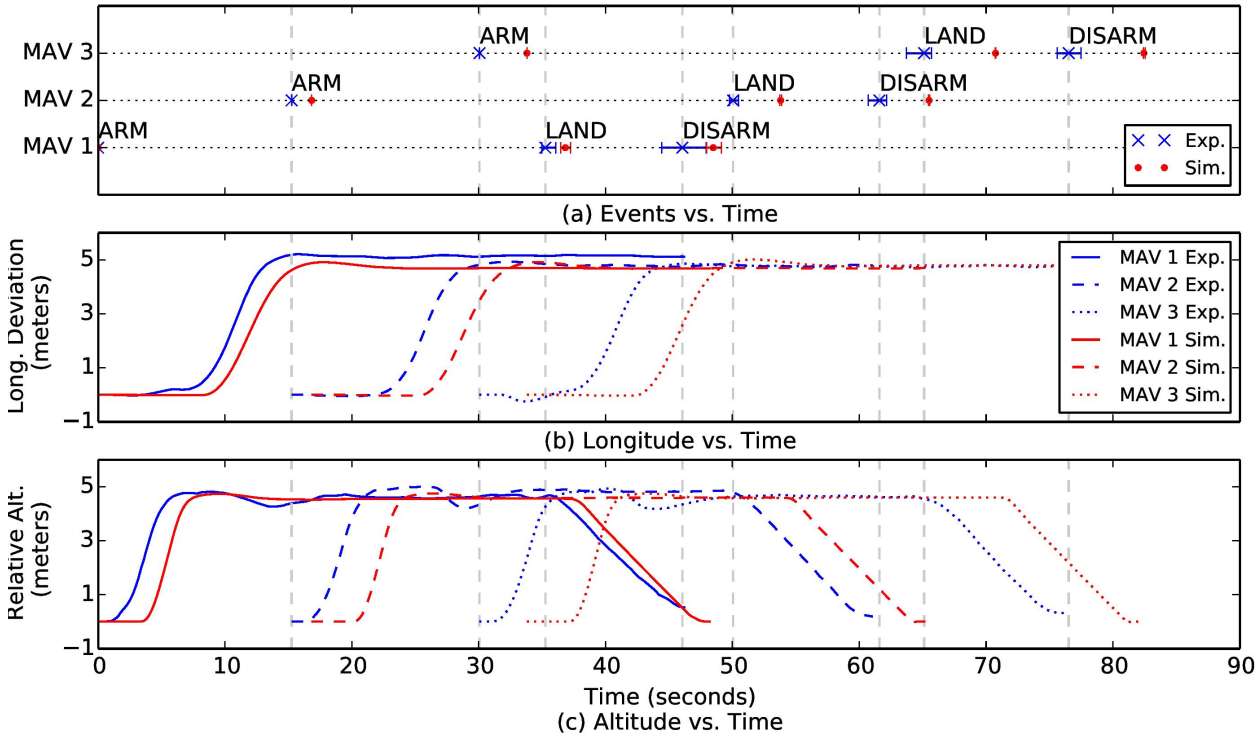


Figure 15: Comparison between emulation and experimentation for a three-drone mission. (a) Events vs. time; (b) Longitude vs. time; (c) Altitude vs. time.

following sub-section. Next, we connected a USRP to our simulator with each agent and used it to communicate between two simulated UAVs to demonstrate extensibility. Finally, we measured energy consumption, speed of flight, and barometric pressure on a real drone and simulated them in the UB-ANC Emulator. This demonstrates the ability of the UB-ANC Emulator to simulate various parameters of interest.

6.4.1.1 Accuracy

To show the accuracy of the UB-ANC Emulator with respect to MAV experiments, we set up three UB-ANC Drones (MAVs) on UB's North Campus to perform a simple "takeoff, loiter, and land" mission. We also execute the same mission in the UB-ANC Emulator. The mission begins when MAV 1 is armed. MAV 1 then takes off to an altitude of 5 meters, flies east for 5 meters, and then loiters (hovers) for 20 seconds before landing. When MAV $i \in \{1, 2\}$ first starts to loiter, it sends a command to MAV $i+1$ to takeoff, loiter, and land in the same pattern. We repeat this three-MAV mission for 5 rounds. Note that, although the UB-ANC drones are capable of completing more sophisticated missions (see, e.g., Section 7), we have selected a relatively simple mission for illustration.

Fig. 15 compares several important quantities that we measured in our experiments and

Approved for Public Release; Distribution Unlimited.

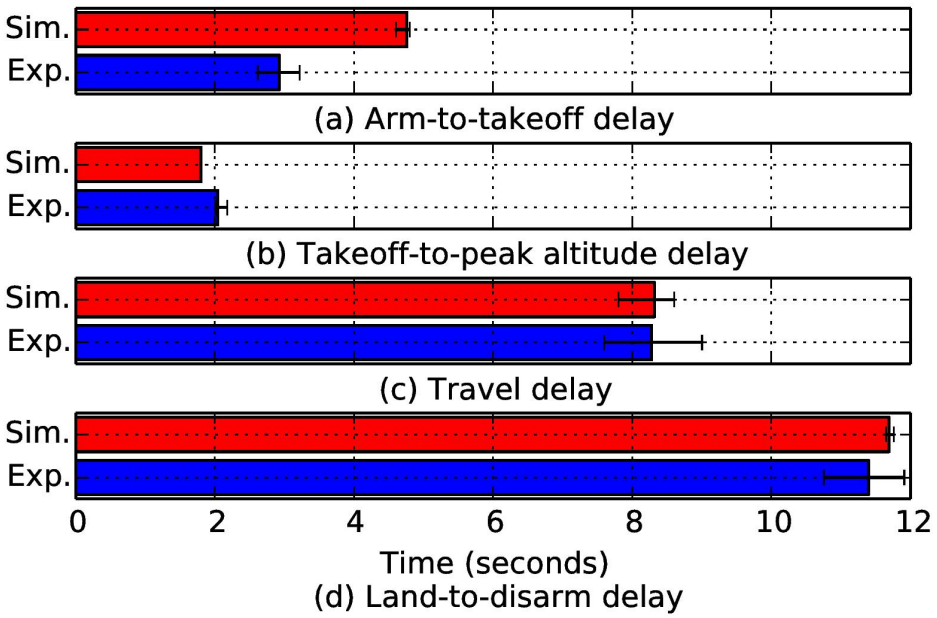


Figure 16: Potential sources of time shift between experiments and simulations for one drone.

simulation. Note that, in Figure 15, time 0 corresponds to the time when MAV 1 is armed. Fig. 15a shows the average (markers) and standard deviation (error bars) of the time at which several key events occur at each MAV (ARM, LAND, and DISARM) over the 5 experimental and simulation rounds. For MAV 1, the ARM event represents the time that it starts to spin-up its motors to takeoff. For MAV $i \in \{2, 3\}$, the ARM event corresponds to a sequence of four events: i) MAV $i-1$ sends a command to MAV i to start its mission; ii) MAV i receives the command from MAV $i-1$; iii) MAV i arms its motors; and iv) MAV i initiates takeoff. For all three MAVs, the LAND event represents the time at which the autonomous landing mode is initiated and the DISARM event represents the time at which the MAV has landed and disarms its motors, signifying the completion of its mission. Fig. 15b and Fig. 15c show each MAV's longitude deviation (from its initial starting position) and relative altitude (from the ground) over time, respectively. Note that we do not show the latitude deviation because it is fixed for the duration of the mission.

It is immediately apparent from Fig. 15(a-c) that there is extra delay in the simulations compared to the experiments. To identify the source of this delay, we have partitioned the MAVs' flight paths into four segments: arm-to-takeoff, takeoff-to-peak altitude, travel, and land-to-disarm. The average and standard deviation of the time required to complete each segment of the flight path for one MAV is plotted in Fig. 16. Clearly, the arm-to-takeoff delay dominates the difference in measured time between simulation and experiment. Therefore, we conclude that it is primarily responsible for the extra delay observed in the simulation results. In private communication with a contributor to the open-source SITL software, we determined that the extra delay is likely due to the fact that the SITL assumes a hexrotor in its parameter definitions, while we are using quadrotors in our experiments. This is near constant in our measurements for each MAV, and can easily be incorporated into the

Approved for Public Release; Distribution Unlimited.

Table 10: Variance of experimental and simulation measurements across 5 rounds.

		MAV 1	MAV 2	MAV 3
Event Var.	**Exp.**	0.5460	0.1365	0.3598
	Sim.	0.1082	0.0010	0.0009
Long. Var.	**Exp.**	0.0138	0.0450	0.0346
	Sim.	0.0133	0.0085	0.0252
Alt. Var.	**Exp.**	0.0281	0.0386	0.0474
	Sim.	0.0119	0.0017	0.0013

Table 11: Mean squared error (MSE) between the average experimental measurements and simulation measurements with (measured) and without (shifted) the extra arm-to-takeoff delay that appears in the simulations.

		MAV 1	MAV 2	MAV 3
Event MSE		2.8070	3.3644	2.8117
Long. MSE	**Measured**	0.3501	0.2286	0.1736
	Shifted	0.1478	0.0085	0.0329
Alt. MSE	**Measured**	0.4678	0.5536	0.5065
	Shifted	0.0463	0.0468	0.0317
Time Shift		1.9937	1.7487	1.8305

simulation based on the exact MAV being used in experiment.

In Table 10, we show the variance of the simulation and experimental data that is shown in Fig. 15. The variance in the simulated event times is negligible, while the variance in the experimental results are reasonably small given the many possible sources of deviation (e.g., wind variations in each round, GPS accuracy, and variation in the flight controller's response to accelerometer and barometer inputs). In Table 11, we show the mean squared error (MSE) between the average experimental measurements and the average simulation measurements with and without compensating for the extra arm-to-takeoff delay that appears in the simulations. We see that, especially when accounting for the time-shift delay, the MSE is quite small. This shows that the emulator provides a good approximation for MAV experiments.

6.4.1.2 Scalability

To show the scalability of the UB-ANC Emulator, we execute a "leader-follower" mission with $N = 25, 50, 75, 100$ MAVs. In this mission, MAV $i+1$ follows 10 meters behind MAV $i \in \{1, 2, \ldots, N-1\}$. This is accomplished by MAV i sending its GPS location to MAV $i+1$ every 100 ms using a 74 byte packet. We execute the leader-follower mission on a Dell Latitude E6530 laptop with an Intel Core-i5 3380M at 2.90 GHz with 2 physical cores (4 logical cores), 16 GB RAM, and running 64-bit Linux Mint 17.3 Cinnamon. Fig. 17 shows the average CPU utilization and memory usage of the UB-ANC Emulator with different numbers of MAVs. We did not observe any noticeable performance degradation except at 100 MAVs, where the GUI exhibited a slow response to inputs (e.g., moving the field of

Approved for Public Release; Distribution Unlimited.

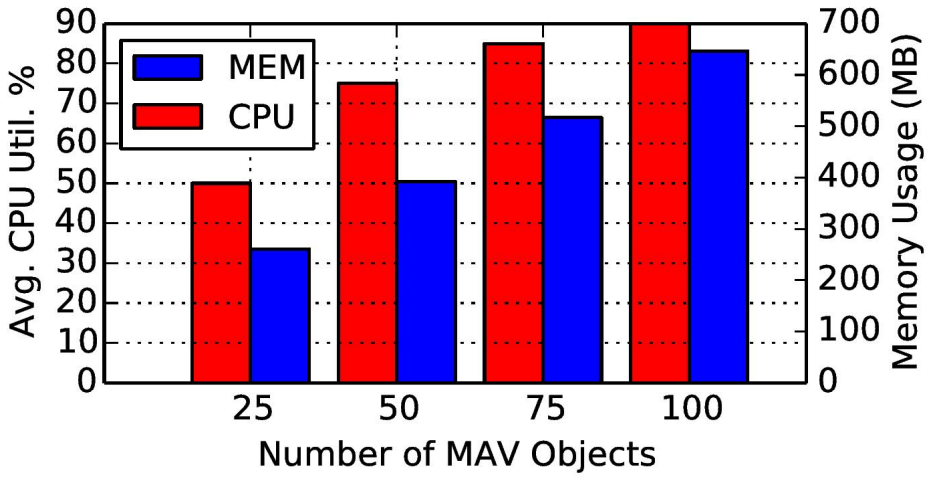


Figure 17: UB-ANC Emulator resource usage for different numbers of emulated MAVs.

view). The interested reader can view a 50-MAV leader-follower mission online.[1]

6.4.1.3 Extensibility

To show the extensibility of the emulator, we set up a follower mission with 5 MAVs. This time, MAV 1 and MAV 2 communicate through two USRP N210 software-defined radios in hardware while the other MAVs communicate in simulation. Fig. 18 shows the setup for this demonstration [8].

6.4.1.4 Simulating Other Parameters

In this section, to demonstrate the usefulness of the emulator, we simulate three other parameters and compare between simulation and experimentation. All of the plots are shown for one of the drones. Fig. 19 shows the average (across five trials) for current and energy consumption. This is useful for energy-sensitive applications. Fig. 20 and Fig. 21 are the plots for flight velocity and sensed pressure change, respectively.

6.4.2 Ns-3 Integration Evaluation

We evaluate the integrated system for two scenarios: node-to-node connectivity (link-level) and network connectivity (end-to-end). For physical layer and channel modeling we use the existing `YansWifiPhy` and `YansWifiChannel` models in ns-3, respectively, which are based on "Yet Another Network Simulator" (YANS) Wi-Fi models [98] for IEEE 802.11b protocols. We generate packet capture (pcap) files using ns-3 and analyze them offline using Wireshark (although other network analyzer tools can also be used).

[1] https://www.youtube.com/watch?v=Qq0cdsofLAA

Approved for Public Release; Distribution Unlimited.

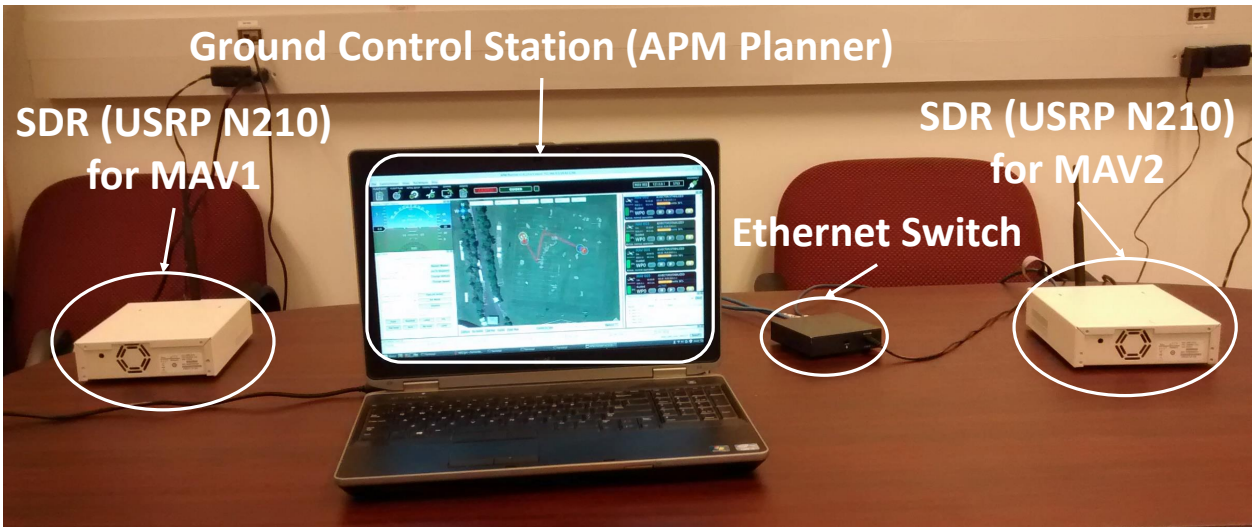


Figure 18: UB-ANC Emulator with two MAVs communicating over USRP N210 software-defined radios.

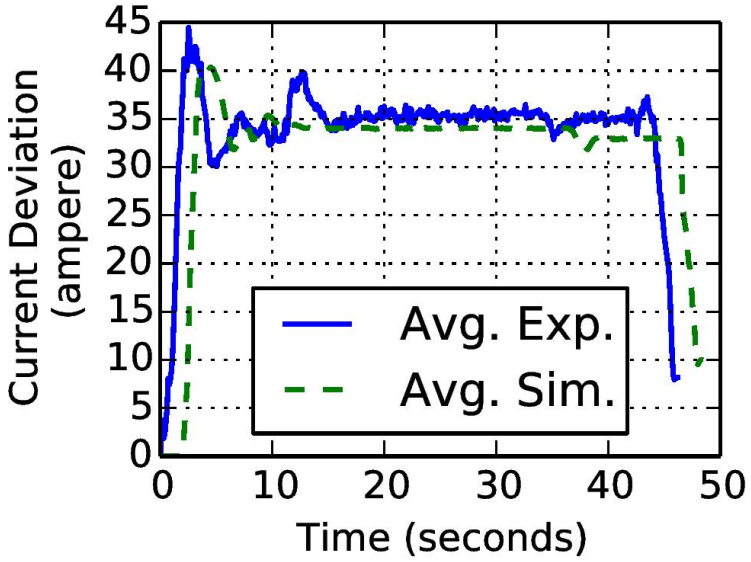


Figure 19: Energy consumption comparison for one drone.

Approved for Public Release; Distribution Unlimited.

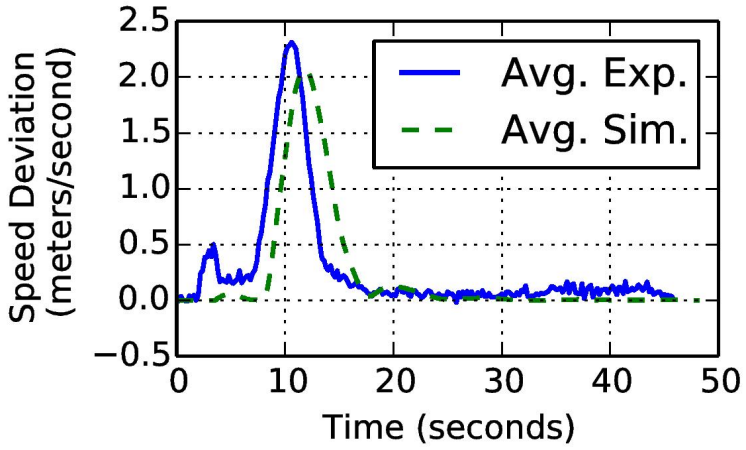


Figure 20: Speed comparison for one drone.

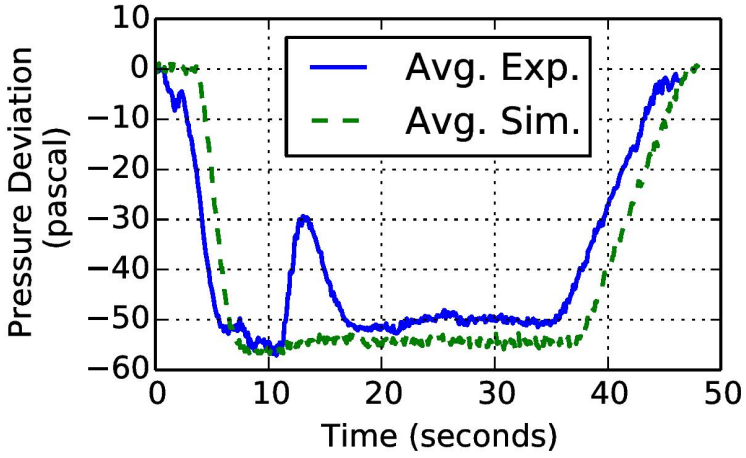


Figure 21: Pressure changes comparison for one drone.

Approved for Public Release; Distribution Unlimited.

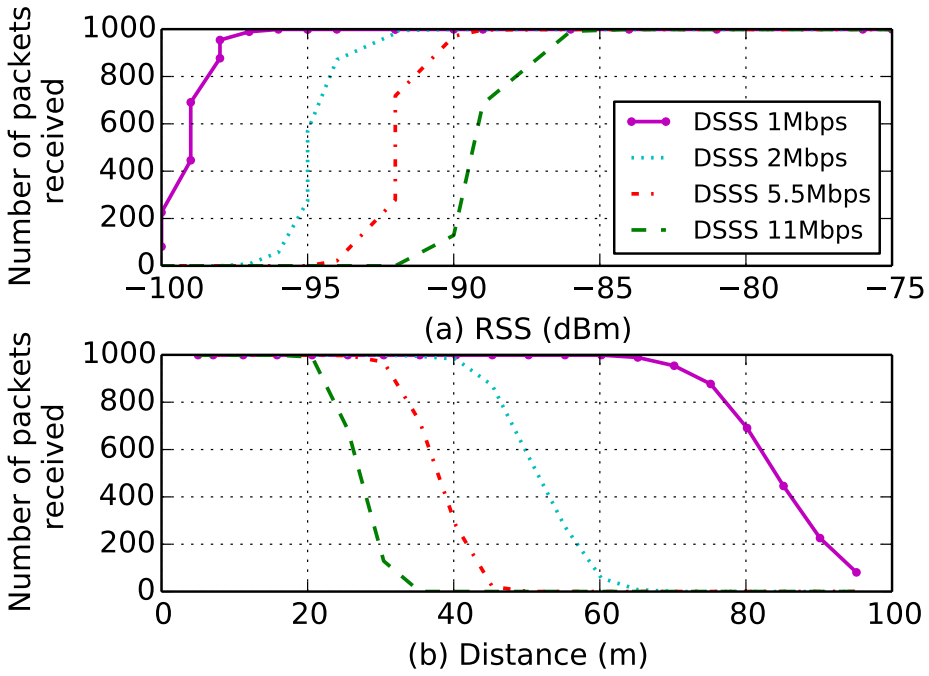


Figure 22: Number of packets received for 1000 packets sent.

6.4.2.1 Node-to-Node Connectivity

In order to evaluate the ns-3 integration at the link-level, we set up a channel measurement mission with two MAVs: MAV1 as the receiver and MAV2 as the sender. When the mission begins, MAV2 takes off to a 5 meter altitude and then hovers while sending 1000 packets at an application (APP) layer rate of 1 packet/s. Each APP layer packet is 7 bytes, which increases to 93 bytes at the MAC layer because of packet headers. After sending 1000 packets, MAV2 flies 5 meters to the east and then hovers while sending another 1000 packets. It repeats this process a total of twenty times during the mission. We run this mission four times using direct sequence spread spectrum (DSSS) modulation with physical (PHY) layer data rates of 1 Mbps, 2 Mbps, 5.5 Mbps, and 11 Mbps. Fig. 22a and Fig. 22b show the number of packets received by MAV1 with respect to the received signal strength (RSS) and 3D Euclidean distance between the two MAVs, respectively. These results closely match those reported in [99] on how Wi-Fi packet reception probabilities vary with RSS in ns-3. This validates our integration and demonstrates the correctness in event and clock synchronization.

6.4.2.2 Network Connectivity (End-to-End)

In order to evaluate the ns-3 integration on end-to-end packet delivery, we set up a network with seven MAVs as shown in Fig. 23 with MAV1 as the source and MAV7 as the destination. We imagine that such an aerial ad-hoc network could be deployed in a disaster situation to enable first responders to communicate when conventional communication infrastructure is down. When the mission starts, all the MAVs take off to a 5 meter altitude, fly to their respective positions, and circle 20 times at a speed of 5 m/s. The centers of adjacent

Approved for Public Release; Distribution Unlimited.

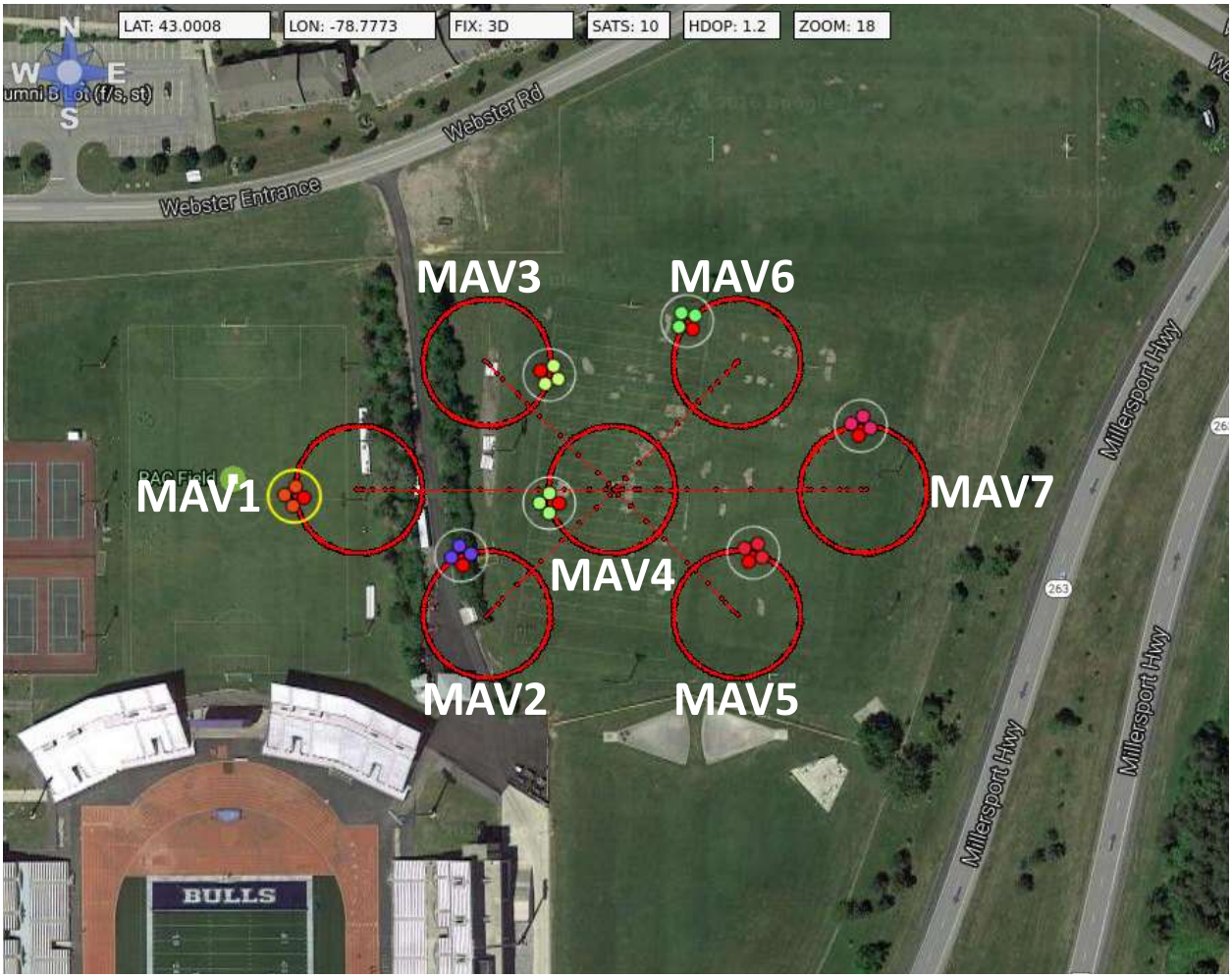


Figure 23: A network of 7 MAVs visualized using APM Planner. MAV1 is the source and MAV7 is the destination.

circles are separated by either $40\sqrt{2}$ m or 80 m and each circle has a 20 m radius. Once it starts circling (approximately 200 seconds into the simulation), MAV1 sends packets to MAV7 (7 bytes at the APP layer; 93 bytes at the MAC layer). We run the test with APP rates of 1 packet/s (low) and 100 packets/s (high) with a PHY rate of 1 Mbps using DSSS for all nodes. Using this network, we compare the performance of two different routing protocols supported by ns-3, namely, Ad hoc On-Demand Distance Vector routing (AODV) and Optimized Link State Routing (OLSR). Fig. 23 shows the emulated network visualized by APM Planner.

Table 12 shows the transmitted and received data rates at the source (MAV1) and destination (MAV7), respectively, excluding overheads. The transmitted data rate at the source is greater under OLSR than AODV. At the same time, the received data rate is lower under OLSR than AODV. Fig. 24 shows the number of packets sent by source (MAV1) and received by the destination (MAV7) over time for different APP rates and different routing protocols. We can clearly see that the transmitted and received data rates vary over time

Approved for Public Release; Distribution Unlimited.

Table 12: Transmitted and received data rates (bytes/s) at MAV1 and MAV7, respectively, excluding overheads.

	Low (1 packet/s)		**High (100 packets/s)**	
	OLSR	**AODV**	**OLSR**	**AODV**
Source (MAV1)	127.71	123.93	11881.94	11735.35
Destination (MAV7)	86.07	89.13	6813.41	8524.34

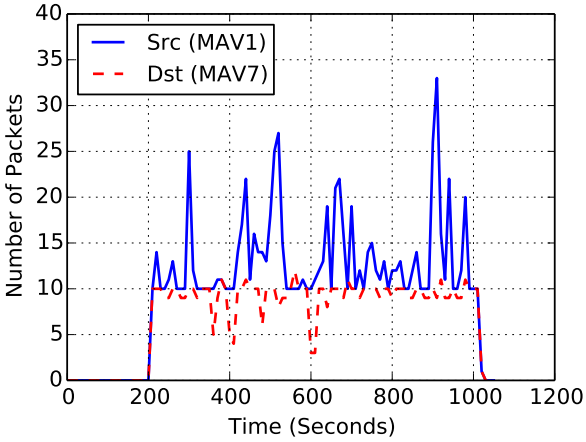
with the MAV's positions and depend heavily on the routing protocol. Fig. 25 shows the total amount of data and routing overheads (in bytes) transmitted by each MAV for different APP rates and routing protocols, showing the unequal traffic load across the MAVs.

6.4.3 Discussion

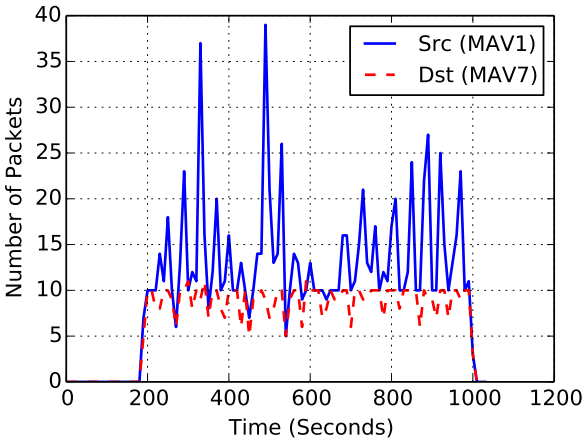
The UB-ANC Emulator aims to make it easy and convenient to design, implement, test, and debug distributed multi-agent mission planning algorithms in software. By integrating it with a network simulator, it can also provide a realistic network environment for evaluating a wide variety of aerial vehicle networking applications. In this section, we described the UB-ANC Emulator's architecture, demonstrated its accuracy with respect to experimentation, and presented a simple API for integrating an existing network simulator into the UB-ANC Emulator and, in particular, showed how this API can be used to integrate ns-3 into the emulator. We used link-level and end-to-end network measurements to verify correctness in event and clock synchronization and demonstrate interaction between the emulator and ns-3. The UB-ANC Emulator is available as open-source at:

https://github.com/jmodares/UB-ANC-Emulator.

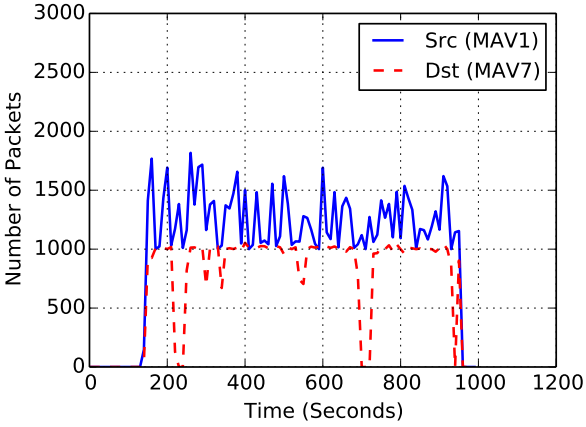
Approved for Public Release; Distribution Unlimited.



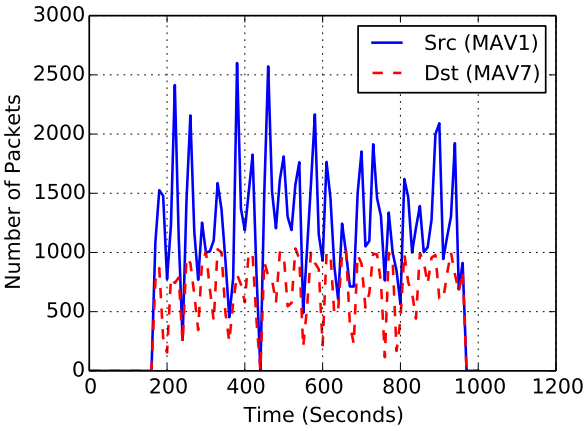
(a) AODV routing with 1 packet/s APP rate.



(b) OLSR routing with 1 packet/s APP rate.



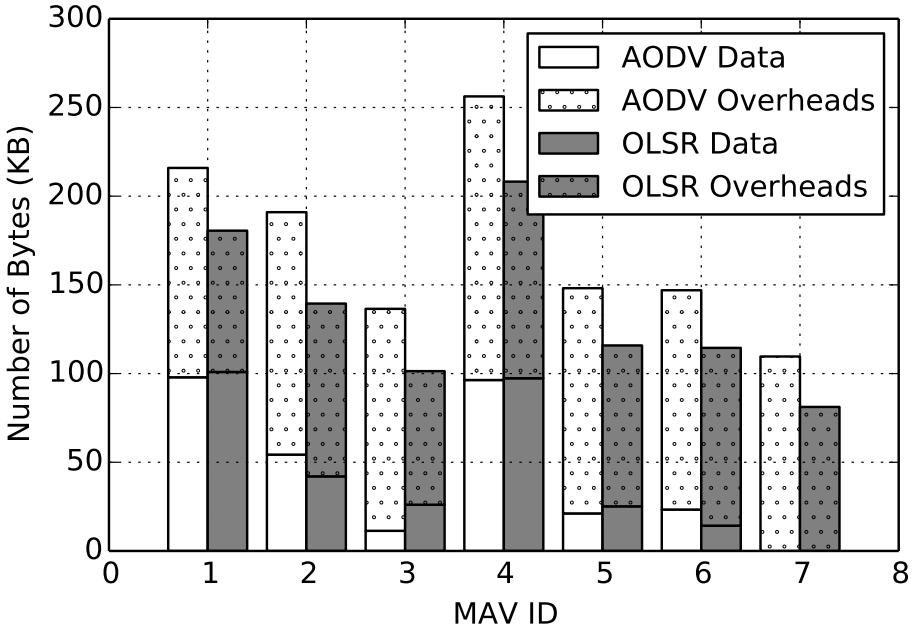
(c) AODV routing with 100 packet/s APP rate.



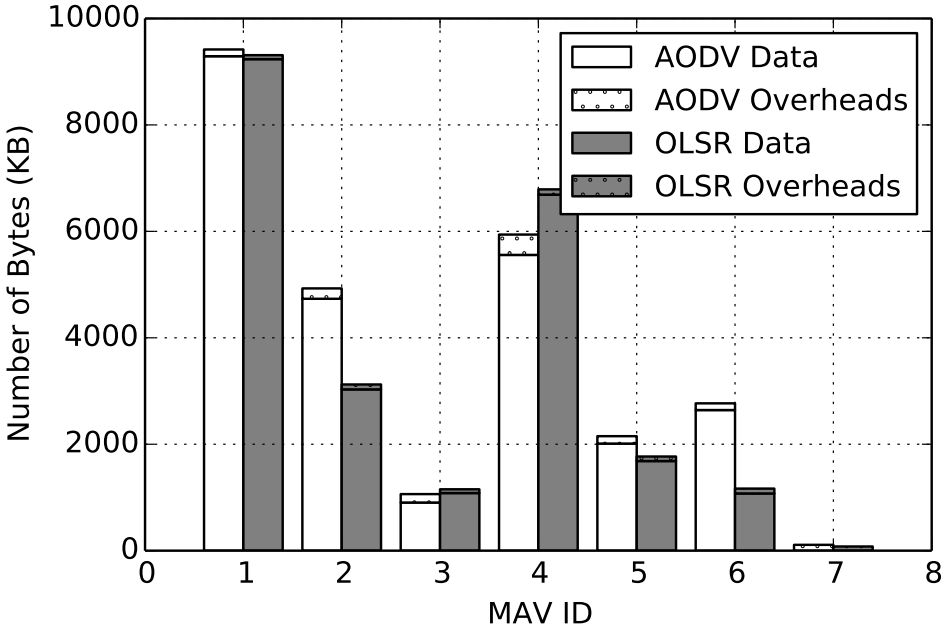
(d) OLSR routing with 100 packet/s APP rate.

Figure 24: Number of data packets sent by MAV1 and received by MAV7 under different APP rates and routing protocols.

Approved for Public Release; Distribution Unlimited.



(a) 1 packet/s APP rate.



(b) 100 packets/s APP rate.

Figure 25: Number of data and routing/overhead packets sent by each MAV under different APP rates and routing protocols.

Approved for Public Release; Distribution Unlimited.

7. *UB-ANC Planner: Energy Efficient Coverage Path Planning with Multiple Drones*

7.1 Introduction

As noted previously, networked unmanned aerial vehicles (UAVs) have emerged as an important technology for public safety, commercial, and military applications including search and rescue, disaster relief, precision agriculture, environmental monitoring, and surveillance. Many of these applications require sophisticated mission planning algorithms to coordinate multiple drones to cover an area efficiently. Such scenarios are complicated by the existence of obstacles, such as buildings, requiring detailed planning for effective operation. Although a lot of work has been done on mission planning, optimal mission planning solutions depend heavily on the specific types of vehicles considered (e.g., ground robots, indoor drones, or outdoor drones), their kinematics, and the specific applications. Prior techniques have been optimized for shortest time to completion or control efficiency. However, a major challenge in the realization of such solutions is the limited energy on each drone.

We consider the problem of covering an arbitrary area containing obstacles using multiple UAVs/drones with a max-min fair energy allocation across drones. Through experimental measurements, we have determined that there are two main factors that affect energy consumption in drones: distance traveled and turns. Traditional coverage path planning algorithms, such as those based on the Traveling Salesman Problem (TSP), are not ideal for drones because they only consider the distance traveled. We present a novel Energy Efficient Coverage Path Planning (EECPP) formulation that explicitly considers the energy consumption characteristics of drones in the path planning optimization, i.e., we not only consider the energy consumed traveling between consecutive waypoints (similar to the TSP), but we also consider the energy consumed by the drone when it accelerates into and out of turns. This work makes four contributions:

- From experimental measurements, we develop a linear model for energy consumption during drone flight.
- Using this model, we formulate the EECPP problem and show that it is NP-hard.
- We decompose the EECPP problem into two sub-problems: a load-balancing problem that fairly divides the area among drones and a minimum energy path planning (MEPP) problem for each drone.

Approved for Public Release; Distribution Unlimited.

- We adapt heuristics proposed for solving the TSP to efficiently solve the MEPP sub-problem on each drone.

The remainder of the section is organized as follows. In Section 7.2, we discuss related work. In Section 7.3.1, we describe our experimental energy measurements and the energy model we derive from them. In Section 7.3.2, we introduce the EECPP problem formulation. In Section 7.4.1, we present our simulation results comparing EECPP to rastering (baseline), depth-limited search, and (where possible) an optimal solution. In Section 7.4.2, we present experimental results comparing our proposed solution to DLS on an actual UB-ANC Drone. We conclude in Section 7.4.3.

7.2 Related Work

Recently, there has been a lot of work on coverage path planning for UAVs. Ahmadzadeh et al. [100] introduce a coverage algorithm for surveillance using a set of fixed-wing UAVs. They utilize dynamic programming to maximize the coverage of the area by a camera mounted on the drone. Maza et al. [101] propose a full coverage algorithm using a set of heterogeneous UAVs (mostly helicopters). First, they generate a polygonal partition of the area that takes into account the capabilities of each individual UAV, such as flight range. Each polygon in the partition is assigned to a UAV that will cover it in a zig-zag pattern (i.e., a raster scan) using a sweep direction that minimizes the number of turns. Environmental obstacles are not considered in [100, 101].

Barrientos et al. [102] present an approach to cover an area using multiple UAVs based on a depth-limited search with back tracking. First, they present a task scheduler to partition the target area into k non-overlapping areas for the k UAVs. The partitioning procedure is based on a negotiation process in which each UAV claims as much area as possible to cover. Then, the wavefront algorithm is used to cover each subarea. Given that their application objective is similar to ours, we have implemented a version of their algorithm and performed comparisons with respect to computational time as well as solution accuracy in the results.

Di Franco et al. [103] discuss an energy-aware coverage path planning solution for a single multi-rotor. They derive energy models for different operating conditions based on real measurements. However, they only consider distance and do not consider the impact of turns in their formulation.

Torres et al. [104] propose a coverage path planning solution for 3D terrain reconstruction with a single UAV. They decompose the area into one or more polygons and have the UAV cover each polygon using a raster scan. They try to minimize the number of turns by calculating the optimal line sweep direction.

Our work is focused on coverage path planning with multiple UAVs. While most previous work attempts to generate paths with minimum distance, our proposed solution instead optimizes energy consumption. Some work has explicitly attempted to minimize turns [101, 102, 104]. However, our empirical results suggest that explicitly optimizing for energy provides more energy-efficient paths than optimizing for turns or distance.

Approved for Public Release; Distribution Unlimited.

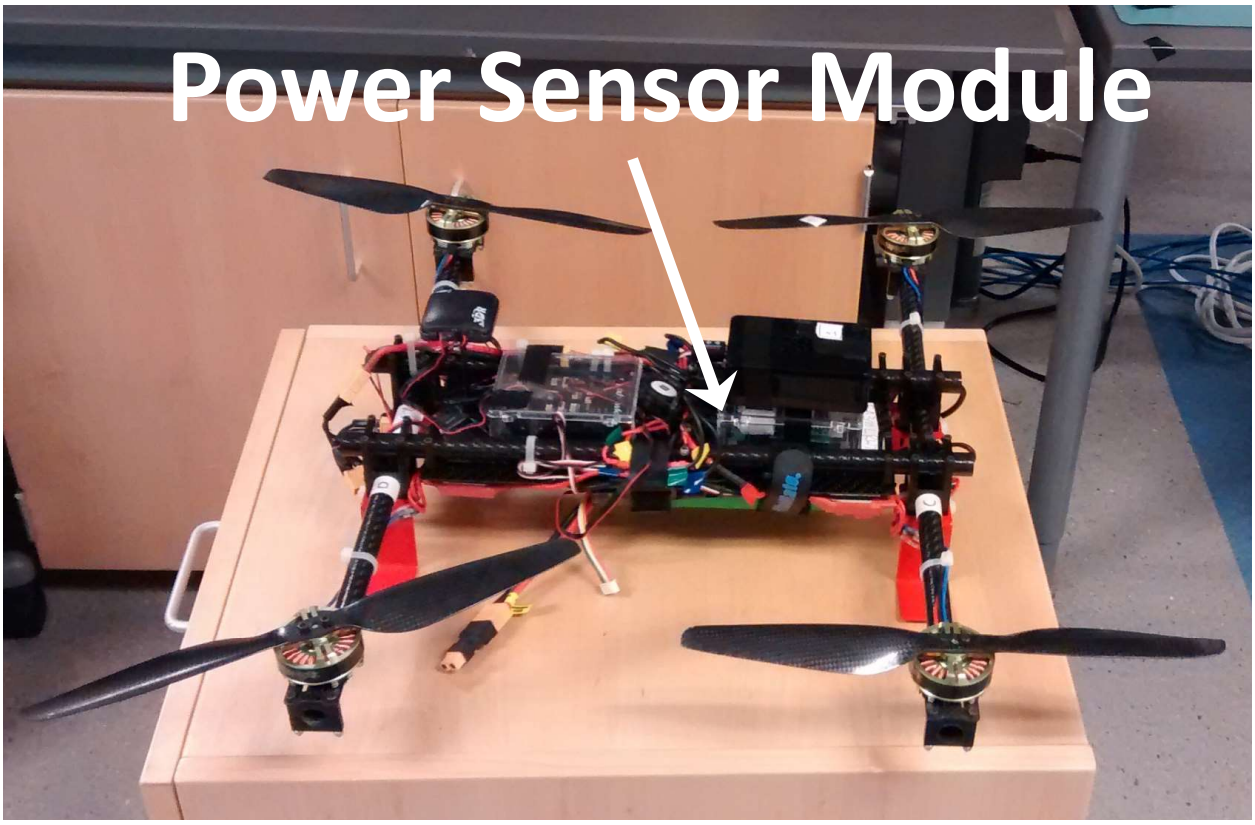


Figure 26: A UB-ANC Drone with a custom frame, waypoint-based Pixhawk flight controller, Raspberry Pi 2, custom power sensor module, and 10,000 mAh battery.

7.3 Methods, Assumptions and Procedures

7.3.1 Energy Consumption of Drone Flight

Consider a realistic scenario where a team of drones is commanded to survey an area. Such an area could contain buildings, towers, and other man-made obstacles, or trees, hills, and other natural ones. Such obstacles can be convex or non-convex, making path planning fairly complex. In addition, path planning for multiple drones to concurrently cover this area is even more challenging. Further, path planning of a single drone could be optimized for several objectives such as shortest time, least distance traveled, least energy used and others. From empirical flight trials, we have concluded that battery energy is the primary resource that limits flight times of such aerial vehicles. Correspondingly, it would be ideal to optimize paths based on energy consumption to enable the drones to cover the maximum area possible.

In order to better understand the power consumption dynamics of the drones, we equipped one of the drones with a power measurement module as shown in Fig. 26. The power measurement module comprises four current sensing modules with ACS712 IC, which translate the passing currents as analog output voltages. We connected the power supply of the 4 motors to the sensors, and mounted all sensors together with an ADC converter with ADS1115

Approved for Public Release; Distribution Unlimited.

IC and a logic level converter. The ADC has four channels, each connected to one sensor, and can send the converted read values via I^2C. The logic level converter lets the IC communicate with the Raspberry Pi on the drone, which has a different logic voltage level.

We used this instrumentation to measure the power consumption of the drone during flight to better understand the relationship between the distance, speed and direction change of the drone and its total power consumption. Each experiment was repeated multiple times and we show averaged results to alleviate anomalies from individual trials.

- **Straight Line Distance:** In this test, we let the drone fly in a straight line with a constant speed of 5 m/s for 3 different distances: 20 m, 40 m, and 60 m. We want to see how the distance traveled affects the power consumption. During flight, the drone takes time to ramp up to the desired velocity and starts slowing down prior to reaching the destination so it can come to a stop at the destination waypoint.

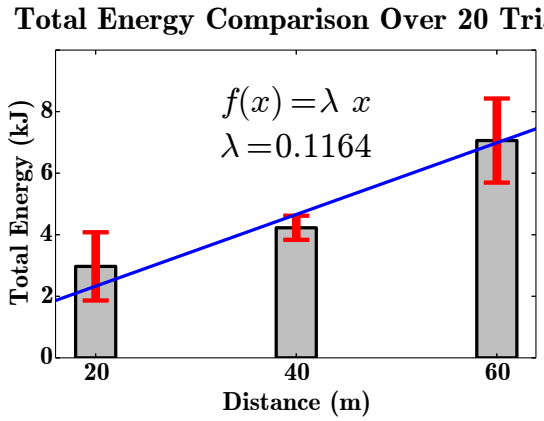
- **Effect of Velocity:** In this test, we let the drone fly in a straight line for a constant distance of $40m$, but with 2 different target speeds: $5m/s$ and $10m/s$. This is to understand the effect of the target flight speed on the power consumption. As mentioned earlier, the drone must ramp up to its target speed and slow down prior to reaching its destination.

- **Effect of Turning:** In this test, we want to observe how direction changes affect power consumption. For this, we flew the drone 40 m with a constant speed of 5 m/s for five turn angles: 0°, 45°, 90°, 135°, and 180°. Specifically, for the 0°turn, we flew a 40 m straight line path and for the other turn angles we flew a 20 m straight line path, turned, and then flew another 20 m straight line path. To isolate the energy consumption associated with turning, we subtracted the average straight path energy consumption from the average total energy that we measured for each angle.

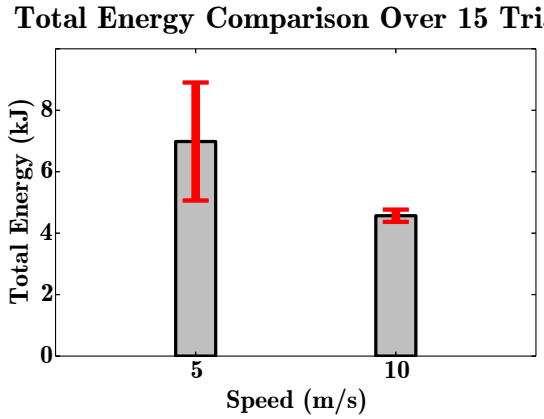
Figs. 27a, 27b, and 27c show the average total energy consumption with standard deviation for the distance, speed, and turn tests, respectively. Fig. 27a shows that increasing the distance traveled increases the energy consumed approximately linearly. Assuming that traveling 0 m incurs 0 energy cost, we performed a linear fit on the measured data and determined that the drone consumes approximately $\lambda = 0.1164$ kJ/m. Fig. 27b shows the relationship between the energy consumption and the target speed. The drone was flown for a distance of 40 m and commanded to fly at the given speed. Lower speeds result in greater time spent by the drone in the air and correspondingly greater energy consumption. Fig. 27c shows the effect of the turn angle on the energy. It is interesting to observe that increasing the turn angle increases the energy consumed for the same distance traveled in an almost linear manner. It is also noteworthy that the variance in energy consumed grows with greater turn angle. We performed a linear fit on the measured data and determined that the drone consumes approximately $\gamma = 0.0173$ kJ/deg. This suggests that intelligently reducing the number and magnitude of turns in a path can potentially reduce energy consumption. Note that we use our measured values of λ and γ when we solve the optimizations proposed in Section 7.3.2.2.

Figs. 28a, 28b, and 28c show the power consumption and flight times for the same tests. All three graphs show that the average power consumption is nearly constant for all traveled

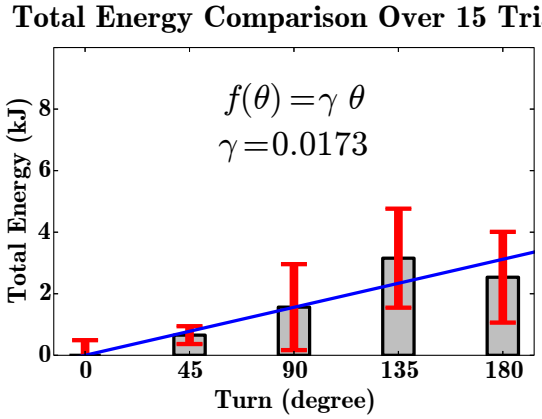
Approved for Public Release; Distribution Unlimited.



(a) Energy vs. Distance



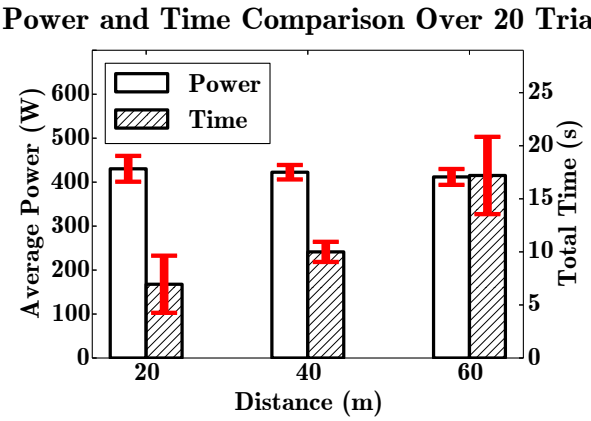
(b) Energy vs. Speed



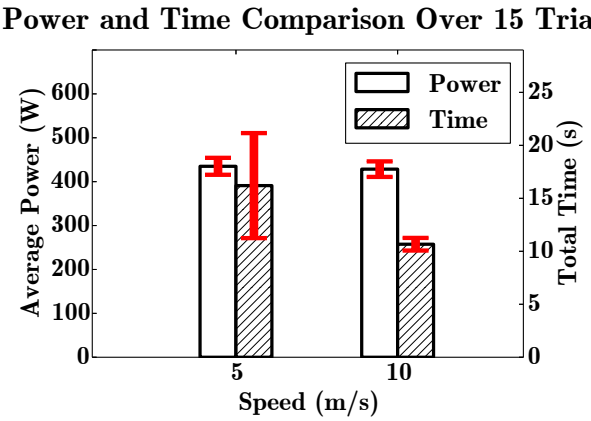
(c) Energy vs. Turn Angle

Figure 27: Energy consumed as measured on the UB-ANC Drone for various patterns of flight. In Fig 27c, we omit the 180^o data point for line fitting as our planning does not allow re-visiting a node. It is shown here to demonstrate model validity.

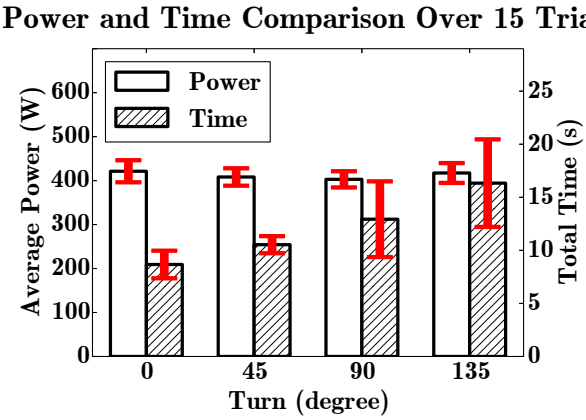
Approved for Public Release; Distribution Unlimited.



(a) Power and Time vs. Distance traveled



(b) Power and Time vs. Speed of travel



(c) Power and Time vs. Turn Angle

Figure 28: Average power draw and time for different missions.

Approved for Public Release; Distribution Unlimited.

distances, speeds, and turn angles. Therefore, the difference in energy consumption across tests is primarily due to the duration of the test. Based on the flight controller's design, the drone often slows down considerably when it enters a turn, which results in an increased flight time (Fig. 28c).

We note that for a different choice of drone, the energy consumption patterns might be different, but we believe that the trends are reasonably indicative of the energy consumption in most drones of this size. Our formulation allows for modeling the energy consumed for each of the path primitives (both distances and angles) and using them as parameters of the optimization. We now develop the EECPP formulation based on our empirical power/energy measurements on drone flight.

7.3.2 Energy Efficient Coverage Path Planning For Multiple Drones

Following from our energy measurements, we formulate the Energy Efficient Coverage Path Planning (EECPP) problem in this section. The problem is divided into two sub-problems: (i) fairly dividing the given area among the drones, and (ii) minimum energy path planning (MEPP) for each drone. Our formulation allows drones to start in different locations and requires each drone to return to its starting point akin to the TSP. These assumptions are drawn from intuition from real applications where surveying is part of a larger operation.

7.3.2.1 Problem Modeling

As our measurements have shown in Section 7.3.1, a drone's energy consumption depends on the distance it travels and the number and degree of turns in its path. To find the minimum energy path for each drone, we formulate a vehicle routing problem (VRP) [105], which is a more general case of the multiple traveling salesman problem (mTSP). The original VRP problem is a min-sum optimization, which tries to minimize the sum of costs over all vehicles. We adapt that to a min-max formulation where we want to minimize the maximum cost incurred (energy expended) by any drone. In literature, this has been referred to as the Newspaper Routing Problem [106], which considers fairness among all vehicles (agents). However, our problem differs from the Newspaper Routing Problem because the objective function not only depends on the distance traveled, but also on the turns.

Surveillance of a given area requires coverage of all locations. However, assuming that the drone is flying at a fixed height, it is able to view a large area from its vantage point. Therefore, we represent the area to be surveyed as a set of grid cells and assume that a grid cell is covered if the drone visits its center. Formally, we represent the grid as a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. We let $i, j, k \in \mathcal{V}$ denote a specific node and $e_{ij} \in \mathcal{E}$ denote an edge between nodes i and j.

We define the pair (α_i, β_i) as the Cartesian coordinate of node $i \in \mathcal{V}$ and let c_{ij} denote the cost of traversing the edge e_{ij} between node $i \in \mathcal{V}$ and node $j \in \mathcal{V}$. In our path planning optimization, we assume that drones traverse adjacent cells. In other words, $e_{ij} \in \mathcal{E}$ if nodes $i, j \in \mathcal{V}$ are adjacent and $e_{ij} \notin \mathcal{E}$ otherwise. Based on our measurements in Section 7.3.1, we

Approved for Public Release; Distribution Unlimited.

assume that the cost (energy) is proportional to the distance traveled: i.e.,

$$c_{ij} = \begin{cases} \lambda\sqrt{(\alpha_i - \alpha_j)^2 + (\beta_i - \beta_j)^2}, & \text{if } e_{ij} \in \mathcal{E} \\ \infty, & \text{otherwise.} \end{cases} \tag{7.1}$$

In other words, if two nodes are adjacent, then the energy cost to traverse the edge between them is proportional to the Euclidean distance between the centers of their corresponding cells (where the parameter λ kJ/m is specific to the drone as described in Section 7.3.1); however, if two nodes are not adjacent, then the cost to traverse them is infinite (i.e., it is not possible to directly traverse the two nodes because they are not connected by any edge).

Let θ_{ijk} denote the exterior angle between nodes $i, j, k \in \mathcal{V}$ (Fig. 29). The squared length of the edges of the triangle made by nodes i, j, k can be determined as follows:

$$r = (\alpha_i - \alpha_j)^2 + (\beta_i - \beta_j)^2, \tag{7.2}$$

$$s = (\alpha_j - \alpha_k)^2 + (\beta_j - \beta_k)^2, \text{and} \tag{7.3}$$

$$t = (\alpha_k - \alpha_i)^2 + (\beta_k - \beta_i)^2 \tag{7.4}$$

We know that given the lengths of three sides of a triangle, we can calculate an internal angle using the Law of Cosines. It follows that the exterior angle between nodes $i, j, k \in \mathcal{V}$ can be written as:

$$\theta_{ijk} = \pi - cos^{-1}\left[\frac{(r + s - t)}{\sqrt{4rs}}\right] \text{ radians.} \tag{7.5}$$

From our empirical energy measurements, we model the cost associated with a feasible turn, denoted by q_{ijk}, to be proportional to the angle of the turn (where the parameter γ kJ/deg is specific to the drone as described in Section 7.3.1):

$$q_{ijk} = \begin{cases} \gamma\frac{180}{\pi}\theta_{ijk}, & \text{if } e_{ij}, e_{jk} \in \mathcal{E} \\ \infty, & \text{otherwise.} \end{cases} \tag{7.6}$$

7.3.2.2 Problem Formulation

Let $\mathcal{A}$ denote the set of agents that will cover the area and let $v_a \in \mathcal{V}$ denote the starting node for agent $a \in \mathcal{A}$. We indicate which edges each agent traverses using the binary decision variable $x_{ij}^a \in \{0, 1\}$, where

$$x_{ij}^a = \begin{cases} 1, & \text{if agent } a \in \mathcal{A} \text{ traverses edge } e_{ij} \in \mathcal{E} \\ 0, & \text{otherwise.} \end{cases} \tag{7.7}$$

Given a feasible path assignment (i.e., a sequence of edges), agent $a \in \mathcal{A}$ will incur a total cost of

$$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}\setminus\{i\}} c_{ij}x_{ij}^a + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}\setminus\{i,v_a\}}\sum_{k\in\mathcal{V}\setminus\{j\}} q_{ijk}x_{ij}^a x_{jk}^a, \tag{7.8}$$

where the first term in the cost function is proportional to the distance traveled (as in the TSP) and the second term is proportional to the sum of turn angles (unique to the EECPP).

Approved for Public Release; Distribution Unlimited.

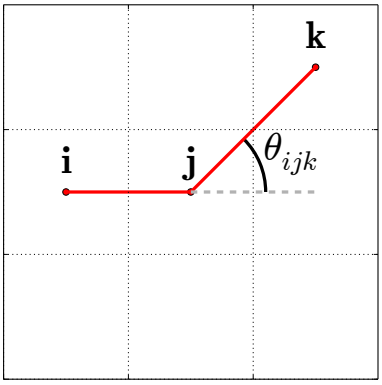


Figure 29: A cell and its neighbors, and the exterior angle for three nodes on the path.

Formally, the EECPP problem can be stated in (7.9). The objective of the EECPP (7.9a) is to determine the paths for each drone that minimize the maximum cost incurred by any individual drone, where the cost function is defined in (7.8). There are several constraints governing this optimization. First, each node should be visited exactly once (7.9b). Next, we need flow conservation constraints which ensure that, once a drone visits a node, it also departs from the same node (7.9c). Third, we incorporate extensions of MTZ-based SECs [105] (subtour elimination constraints) to a three-index model (7.9d).[1] Finally, u_i is a dummy variable associated with node $i \in \mathcal{V}$ which is assigned by the solver.

$$\min \max_{a \in \mathcal{A}} \quad \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V} \setminus \{i\}} c_{ij} x_{ij}^a + \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V} \setminus \{i, v_a\}} \sum_{k \in \mathcal{V} \setminus \{j\}} q_{ijk} x_{ij}^a x_{jk}^a \tag{7.9a}$$

$$\text{s.t.} \quad \sum_{a \in \mathcal{A}} \sum_{i \in \mathcal{V} \setminus \{j\}} x_{ij}^a = 1, \ \forall j \in \mathcal{V} \tag{7.9b}$$

$$\sum_{i \in \mathcal{V} \setminus \{j\}} x_{ij}^a - \sum_{k \in \mathcal{V} \setminus \{j\}} x_{jk}^a = 0, \ \forall a \in \mathcal{A}, \forall j \in \mathcal{V} \tag{7.9c}$$

$$u_i - u_j + |\mathcal{V}| \, x_{ij}^a \leq |\mathcal{V}| - 1, \quad \forall a \in \mathcal{A}, \forall i, j \in \mathcal{V} \setminus \{v_a\} \text{ and } i \neq j \tag{7.9d}$$

$$u_i \in \mathcal{Z}, \ \forall i \in \mathcal{V} \tag{7.9e}$$

$$x_{ij}^a \in \{0, 1\}, \ \forall i, j \in \mathcal{V} \text{ and } \forall a \in \mathcal{A} \tag{7.9f}$$

The problem shown above is an NP-hard mixed integer quadratic constrained program (MIQCP); therefore, it does not scale well beyond a few drones and dozens of cells. To overcome this limitation, we decompose the problem into two sub-problems: the first sub-problem assigns a set of cells to each drone and the second determines the minimum energy path that each drone will follow to cover these cells.

[1] A subtour is a closed path that starts from one node and returns to that node. The subtour elimination constraints prevent the optimization solver from returning undesirable subtours as solutions.

Approved for Public Release; Distribution Unlimited.

Sub-Problem 1: Load Balancing (LB) The first sub-problem is the so-called load balancing (LB) problem. We use mixed integer linear programming (MILP) to divide the nodes among the users with linear complexity. Let c_{ai} denote the distance between agent $a \in \mathcal{A}$ and node $i \in \mathcal{V}$. Let x_{ai} denote a decision variable that is set to 1 if node $i \in \mathcal{V}$ is assigned to agent $a \in \mathcal{A}$ and is set to 0 otherwise: i.e.,

$$x_{ai} = \begin{cases} 1, & \text{if node } i \in \mathcal{V} \text{ is assigned to agent } a \in \mathcal{A} \\ 0, & \text{otherwise.} \end{cases} \tag{7.10}$$

Based on the starting positions of the drones, we would like to assign grid cells to them to minimize the maximum energy incurred across them. This can be formulated as:

$$\min \max_{a \in \mathcal{A}} \quad \sum_{i \in \mathcal{V}} c_{ai} x_{ai} \tag{7.11a}$$

$$\text{s.t.} \quad \sum_{a \in \mathcal{A}} x_{ai} = 1, \ \forall i \in \mathcal{V} \tag{7.11b}$$

$$x_{ai} \in \{0, 1\}, \ \forall a \in \mathcal{A} \text{ and } \forall i \in \mathcal{V} \tag{7.11c}$$

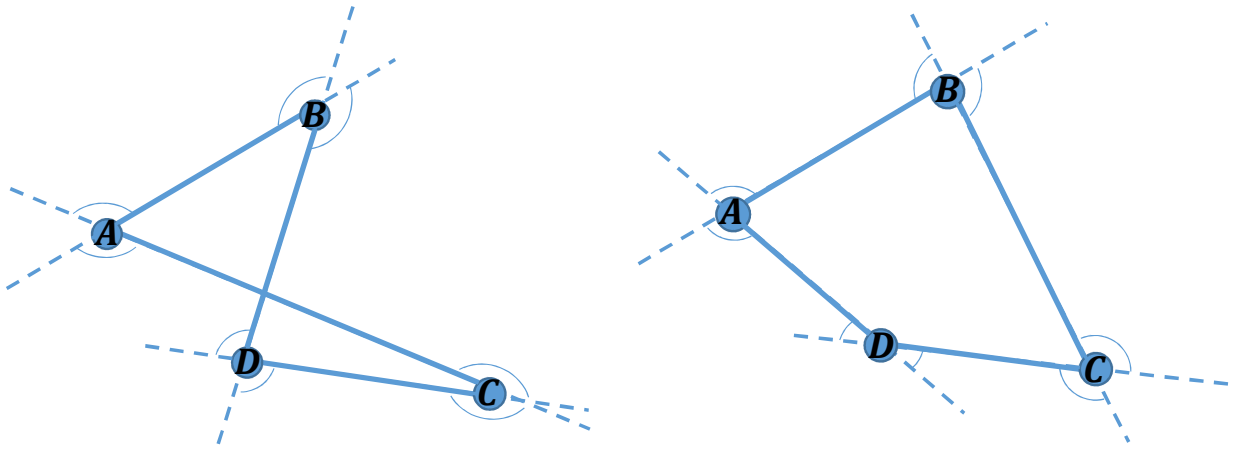
This can be solved by a linear program with complexity that is linear in the product of the number of drones and the number of grid cells.

Sub-Problem 2: Minimum Energy Path Planning (MEPP) The second sub-problem is the minimum energy path planning (MEPP) problem. After dividing the area in sub-problem 1, we use mixed integer quadratic programming (MIQP) to formulate the MEPP problem. The problem is very similar to the EECPP (7.9a) except that there are no indices for individual drones and it is shown below.

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V} \setminus \{i\}} c_{ij} x_{ij} + \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V} \setminus \{i, v_a\}} \sum_{k \in \mathcal{V} \setminus \{j\}} q_{ijk} x_{ij} x_{jk} \\ \text{s.t.} \quad & \sum_{i \in \mathcal{V} \setminus \{j\}} x_{ij} = 1, \ \forall j \in \mathcal{V} \\ & \sum_{j \in \mathcal{V} \setminus \{i\}} x_{ij} = 1, \ \forall i \in \mathcal{V} \\ & u_i - u_j + |\mathcal{V}| \, x_{ij} \leq |\mathcal{V}| - 1, \\ & \quad \forall i, j \in \mathcal{V} \setminus \{v_a\} \text{ and } i \neq j \\ & u_i \in \mathcal{Z}, \ \forall i \in \mathcal{V} \\ & x_{ij} \in \{0, 1\}, \ \forall i, j \in \mathcal{V} \end{aligned}$$

The above minimum energy path planning problem is NP-hard since it is similar to the TSP, but with additional quadratic terms to account for the turning costs. To solve this problem efficiently when a large number of nodes are assigned to a drone, we propose a modification of the well-known Lin-Kernighan Heuristic (LKH [107]). While the conventional LKH only considers the distance traveled, we modify it to also account for the drone's turning costs.

Approved for Public Release; Distribution Unlimited.



(a) Initial feasible tour T. (b) New feasible tour T' obtained from tour T.

Figure 30: Illustration of the Lin-Kernighan Heuristic (LKH).

Lin-Kernighan Heuristic for Drones (LKH-D): We now briefly describe the LKH used to solve the TSP. We then show how we have modified the LKH to account for the cost of turns in the MEPP problem. We call the new algorithm LKH for Drones (LKH-D).

LKH begins by determining a feasible tour T that visits each node exactly once and returns to the origin node. The tour T is associated with a cost $f(T)$, which is equal to the length of the tour. LKH works by iteratively improving the initial tour using a specific transformation (see, e.g., [107]). After applying the transformation on the tour T, a new feasible tour T' is obtained, which has a cost $f(T')$. If the gain $g(T, T') = f(T) - f(T')$ is positive (i.e., the tour T' has a lower cost than the tour T), then the new tour is adopted; otherwise, it is thrown away. This procedure is repeated iteratively until a specific stopping condition is met (see, e.g., [107]). Fig. 30a and Fig. 30b show a four node tour T and a feasible tour T' obtained by an appropriate transformation on T: in this example, T' is obtained from T using a "flip" operation that replaces edges $\overline{BD}$ and $\overline{CA}$ with edges $\overline{BC}$ and $\overline{DA}$, respectively.

Our proposed LKH-D algorithm follows the same iterative approach as the conventional LKH, but uses a different cost function to account for the drone's turning costs. Specifically, $f(T)$ is calculated as a weighted sum of the length of the tour and the sum of the turn angles within the tour (these are the four exterior angles illustrated in Figs. 30a and 30b.) In other words, $f(T)$ is equal to the energy cost defined in (7.8) associated with the tour T.

7.4 Results and Discussion

7.4.1 Simulation Results

We perform several sets of simulations to demonstrate the benefits of our proposed heuristics for the EECPP problem with multiple drones and the MEPP problem (i.e., sub-problem 2) for a single drone. We compare our solution with (a) simple rastering, which has no planning cost, (b) a previously proposed depth-limited search (DLS) algorithm with backtracking for

Approved for Public Release; Distribution Unlimited.

multi-robot search [102],[2] and (c) an optimal solution wherever possible. (We imposed a 1 hour execution time limit for all algorithms, and present the best result obtained in that time). We have compared these algorithms in several scenarios to demonstrate the utility of our solution. For all our simulations, we use the UB-ANC Emulator introduced in Section 6.

We perform comparisons with the benchmark algorithms in several dimensions. We vary the area of coverage by a single drone to compare the scalability of the algorithms with increased area (number of grid cells). We compare the efficiency of each approach by comparing the total energy consumed by the drone for that mission. We also perform these comparisons in four scenarios with one or more obstacles as shown in Fig. 32. Finally, we demonstrate a full run of our multi-robot path planning problem by simulating a set of drones covering a large area. Our results are from simulations on a standard laptop. We impose a 1-hr run limit on all our algorithms.

We note that Sections 7.4.1.1 and 7.4.1.2 focus primarily on the MEPP problem for a single drone. In Section 7.4.1.3, we consider the full EECPP problem with multiple drones covering a large area.

7.4.1.1 Algorithm Scalability

First, we show how the compared algorithms perform in terms of computation time and energy efficiency for simple rectangular maps with dimensions 2×4, 3×6, 4×8, and 5×10. The cells in each of these maps are 10 m $\times$ 10 m squares and there are no obstacles. Fig. 31a shows how the computation time scales with the number of cells for each algorithm and Fig. 31b shows the energy consumption under each algorithm. Please note that the time axis in Fig. 31a is in log-scale. As would be expected, the optimal solution for the MEPP problem, which is similar to the TSP, is computationally expensive. Rastering does not require any planning and is not represented in the figure. In comparison to DLS [102], the proposed LKH-D is three orders of magnitude faster for the large map. Fig. 31b shows that for small areas without obstacles all algorithms achieve comparable performance to the exhaustive CPLEX solution (which is optimal for grid sizes up to 4×8).

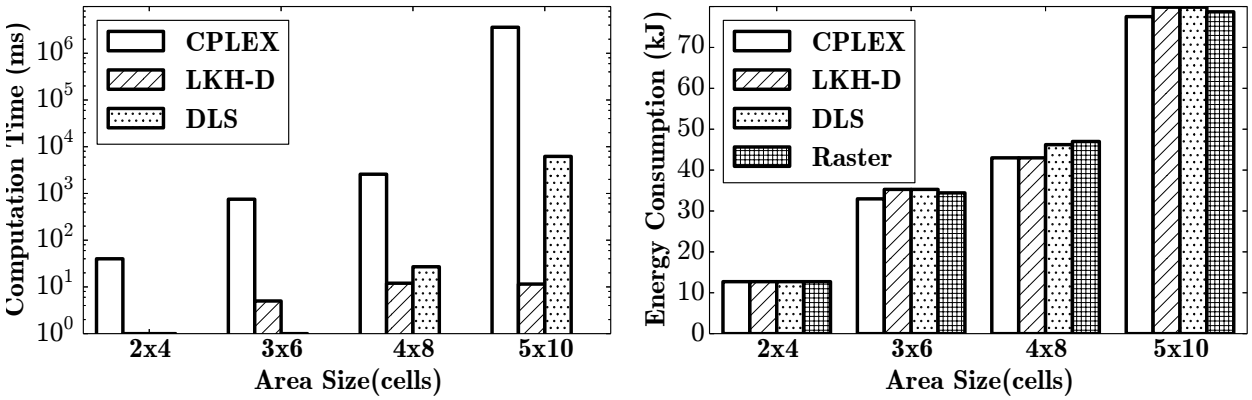
7.4.1.2 Energy Efficiency and Algorithm Adaptivity

As discussed in the introduction, a major challenge in path planning is adapting to real-world constraints such as obstacles and areas shaped in a non-standard manner. To understand the effects of obstacles on the computation time and performance of the compared algorithms, we generated four 8×15 rectangular maps as illustrated in Fig. 32. Given the size of the area (120 cells), we were unable to run CPLEX to solve the MEPP problem to completion in all cases. Instead, we run the optimization for one hour and report results returned by the CPLEX optimization solver.

Fig. 33a shows the computation time for the compared algorithms when they are applied to the four areas defined in Fig. 32, and Fig. 33b shows the corresponding energy consumption. In all scenarios, our proposed heuristic (LKH-D) performs at least as well as the time-limited CPLEX solution and does approximately 15-20% better than the DLS and

[2]Since no source code was available for this algorithm, we have faithfully implemented our version of the proposed algorithm and verified its functionality with results from various papers on this algorithm.

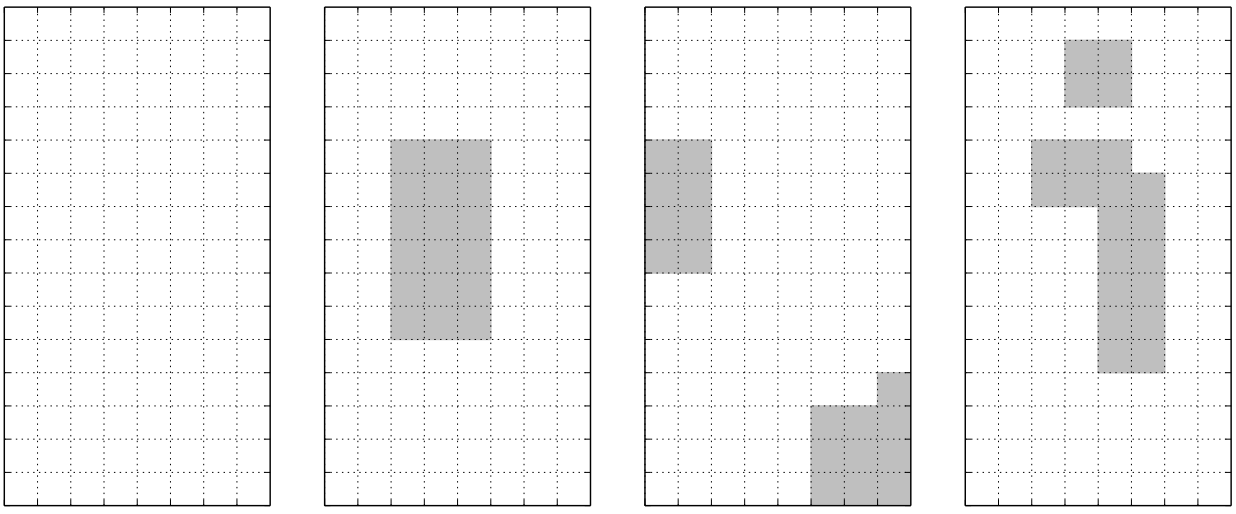
Approved for Public Release; Distribution Unlimited.



(a) Computation time

(b) Performance

Figure 31: Run-time (max: 1 hr) and performance of MEPP on rectangular grids of different sizes without obstacles.



(a) Area 1

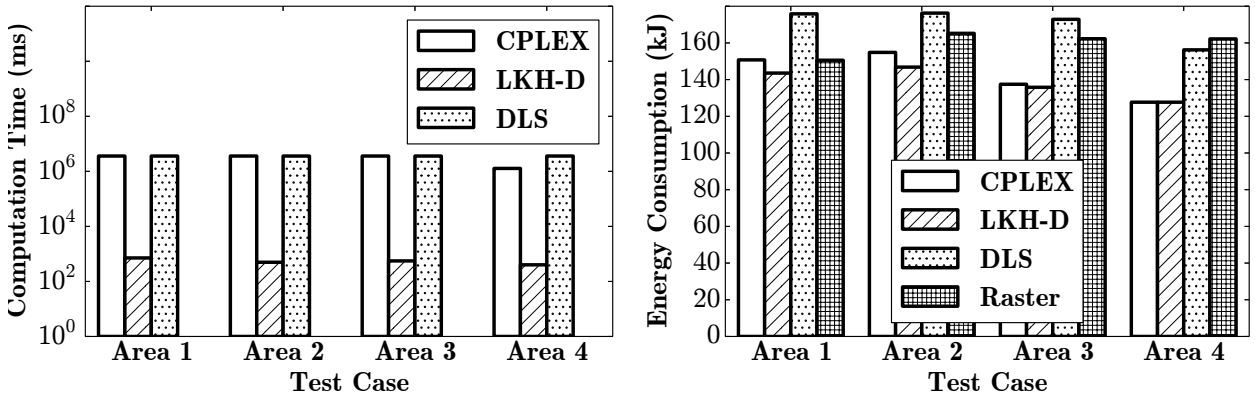
(b) Area 2

(c) Area 3

(d) Area 4

Figure 32: Rectangular grid maps used to evaluate the algorithms. Grey cells represent obstacles.

Approved for Public Release; Distribution Unlimited.



(a) Computation time

(b) Performance

Figure 33: Comparison of algorithms over areas in Fig. 32.

rastering solutions. Moreover, LKH-D achieves this performance in 3-orders of magnitude less time than we allowed for the CPLEX and DLS solutions. These results highlight the benefits of both our modeling and the proposed heuristic for energy-efficient path planning.

Fig. 34 illustrates the planned paths under each algorithm for the area shown in Fig. 32d, and Table 13 provides some statistics about these paths (energy, computation time, total distance, and total degree of turns). The proposed heuristic (LKH-D) achieves the optimal energy consumption, the minimum distance traveled, and the minimum total turns, and is executed in less than half a second.

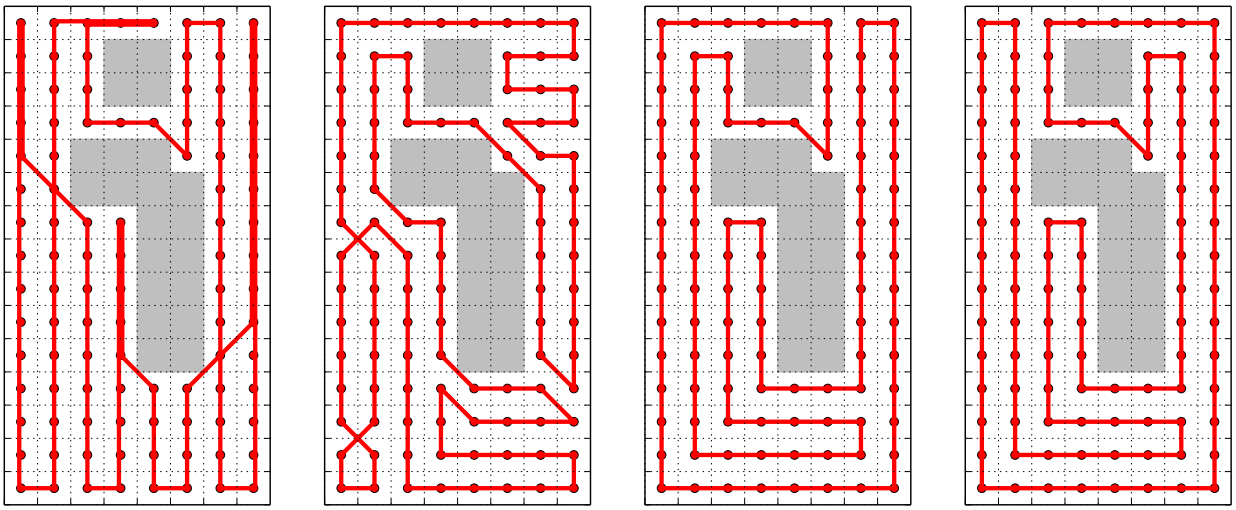
7.4.1.3 Multi-Drone Path Planning

To show the application of the proposed method for solving the EECPP problem (load balancing followed by LKH-D to solve the MEPP) in a large-scale scenario, we executed it with a varying number of drones (5, 10, 15, and 20) on the University at Buffalo's North Campus in simulation. A top view of the area can be seen in Fig. 35. The area is decomposed into over 3000 cells measuring 20 m $\times$ 20 m. Running the multi-agent coverage path planning on the area with different numbers of drones, with both LKH and LKH-D algorithms, we obtain the results shown in Fig. 36a and 36b. Fig. 36a compares them with respect to average computation time required for the path planning per drone across all drones in each mission. Fig. 36b compares these algorithms with respect to average energy expended per drone. As expected, increasing the number of available drones allows each agent to cover less area, and correspondingly decreases the computation time and energy expended for the mission. Finally, LKH-D achieves lower energy consumption than LKH, at the expense of increased computation time, because it considers the turning costs.

7.4.2 Experimental Evaluation

To recap, we have now formulated the EECPP problem and shown that it is NP-hard. We then split it into two problems: (i) decomposing the search area among the available drones

Approved for Public Release; Distribution Unlimited.



(a) Rastering (b) DLS (c) CPLEX (d) LKH-D

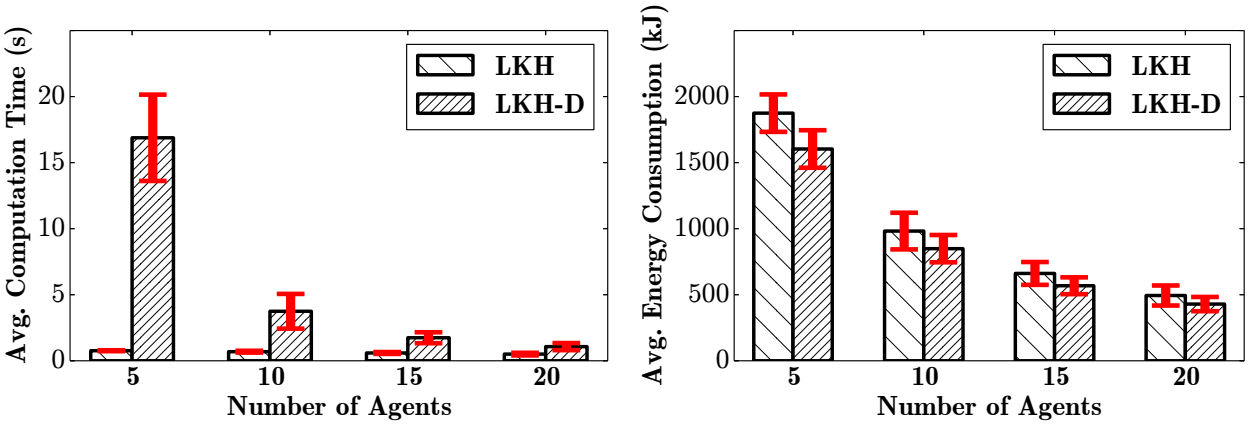
Figure 34: Visualization of the planned paths for different algorithms in area 4 of Fig. 32. The drone's tour starts and ends in the upper-right corner of the grid.

Table 13: Simulation statistics for area 4 in Fig. 32.

Algorithm	Raster	DLS (1 hr)	CPLEX	LKH-D
Energy (kJ)	162.2	156.2	127.7	127.7
Comp Time (s)	0	3600	1276	0.4
Distance (m)	1244.8	1043.8	994.1	994.1
Turns (degree)	1620	2970	1620	1620

Approved for Public Release; Distribution Unlimited.

Figure 35: UB North Campus. Areas dense with buildings are assumed to obstacles (shaded with diagonal lines).



(a) Computation time

(b) Performance

Figure 36: Average path planning computation time (per drone), and energy consumption (per drone) for a set of drones covering UB North campus. Comparison between LKH and LKH-D algorithms.

Approved for Public Release; Distribution Unlimited.

Table 14: Experimental statistics for Area 4 in Fig. 32.

Algorithm	**DLS (1 hr)**	**LKH-D**
Energy (kJ)	111.1	83.9
Distance (m)	1044.1	996.4
Flight Time (s)	391.6	293.7
Average Speed (m/s)	2.66	3.39

(LB); and (ii) planning energy-efficient paths for each individual drone (MEPP). We showed that the first problem was solvable in linear time while the second one was a more complex version of the Traveling Salesman Problem (TSP) and therefore NP-hard. We demonstrated via simulation that our heuristic is computationally more efficient than CPLEX's branch-and-bound algorithm or the previously proposed DLS algorithm. Our LKH-D heuristic was also better in energy efficiency. We wanted to ensure that these results held true in actual flight tests as well.

To this end, we cover the UB stadium using one UB-ANC Drone with a target speed of 5 m/s, first with the DLS algorithm and then with our LKH-D approach. We include virtual obstacles mimicking Area 4 in Fig. 32 so that the planned coverage paths are the same as shown in Fig. 34. Since the planning algorithm generated missions are in standard format, we deployed them directly onto the drone. The planned missions using DLS and LKH-D opened in the APM Planner application are shown in Figs. 37a and 38a, respectively. The drone starts its mission from the top right corner of the field, following the planned path and returning to the starting point.[3] The result of the coverage is shown in Table 14. Our LKH-D approach improves the overall energy consumed for the coverage by 25% compared to the DLS algorithm. This matches well with the simulation results in Table 13 (Cfg. B), which show that LKH-D achieves approximately 22% lower energy consumption than DLS.

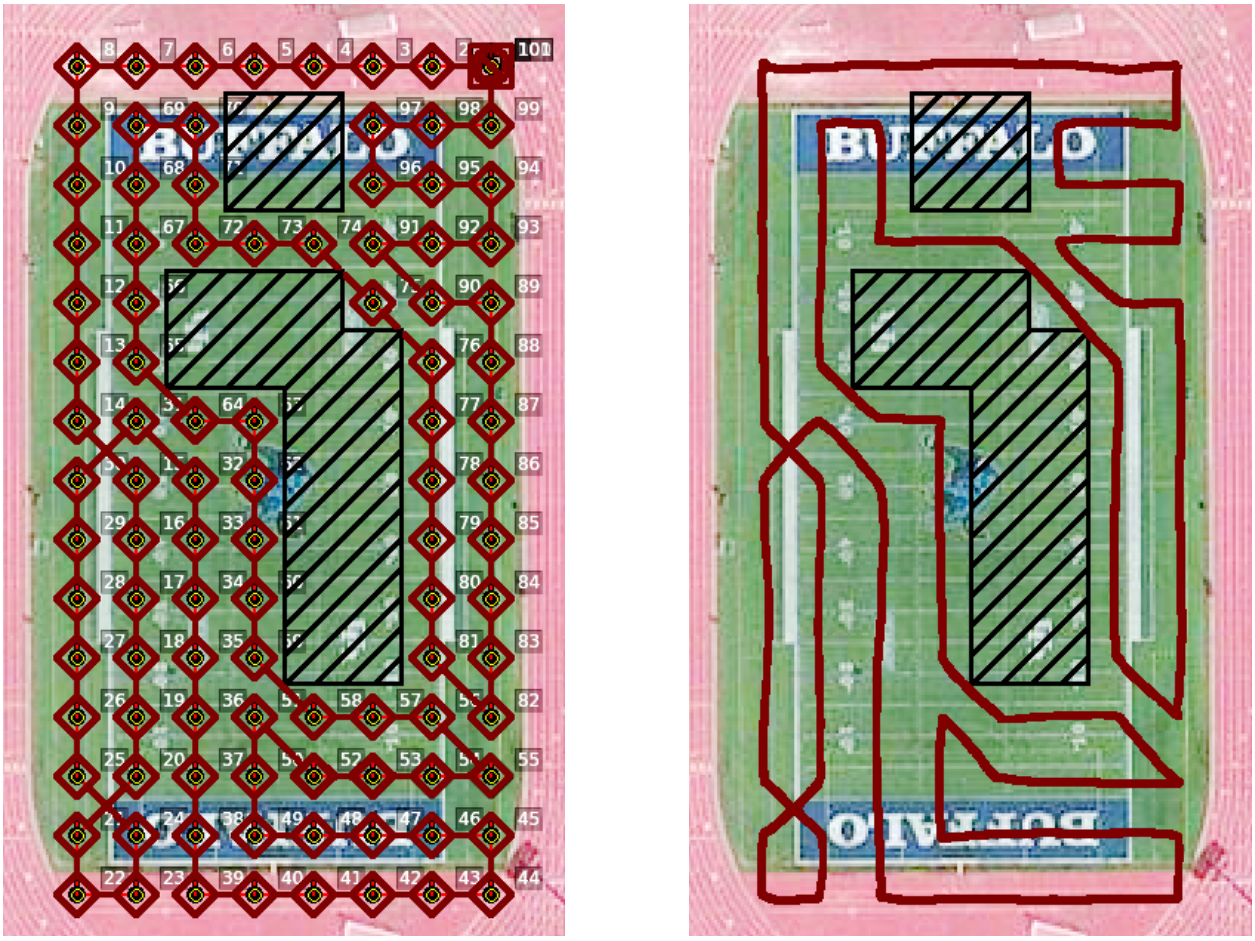
We note that LKH-D determines a more efficient flight path with less turns than DLS, which allows the drone to cover the area at a higher average speed (because it does not need to slow down as frequently for turns), reduces the time that it requires to cover the area, and ultimately reduces its energy consumption.

7.4.3 Discussion

We formulated the Energy Efficient Coverage Path Planning (EECPP) problem for covering an arbitrary area containing obstacles using multiple drones. The goal of the EECPP problem is to minimize the maximum energy required for any individual drone to traverse its assigned path. Unlike the conventional multiple traveling salesman problem (mTSP), the vehicle routing problem (VRP), and the newspaper routing problem, which only consider the distance traveled by each agent, we accounted for the energy consumption characteristics of drones in our optimization. In particular, we included a term to account for the additional energy consumed by drones when they accelerate into and out of turns. This decision was

[3]We note that, for our LKH-D experiments, we inadvertently started the drone just north of its programmed starting/ending position. Consequently, its actual starting and ending positions do not match and the experimental performance is slightly degraded due to the extra distance traveled.

Approved for Public Release; Distribution Unlimited.

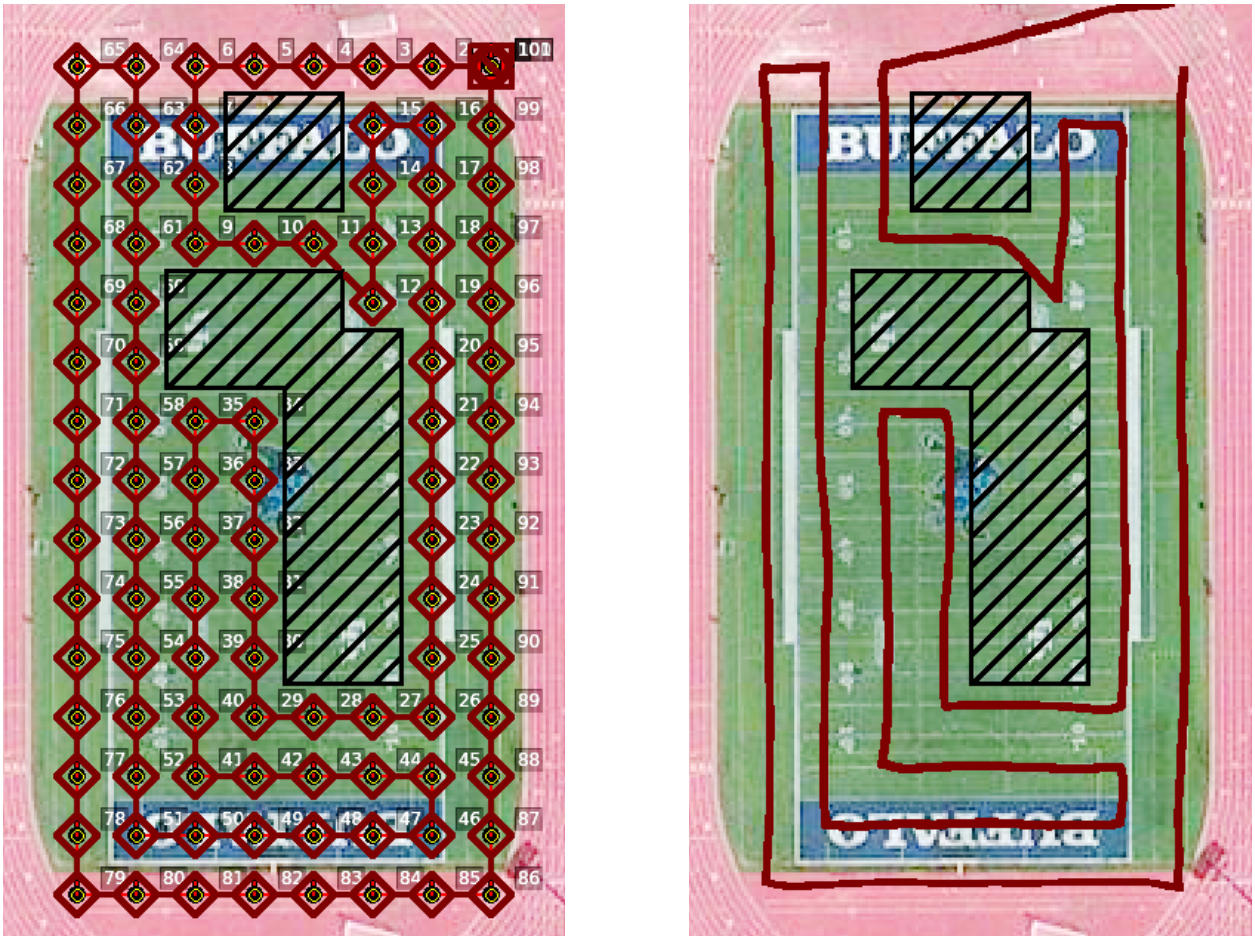


(a) Path made by DLS (Target)

(b) Path made by DLS (Experiment)

Figure 37: Satellite view from UB stadium and the planned coverage paths. Virtual obstacles are shaded with diagonal lines. (a) Path planned by DLS algorithm. (b) Actual flight path in experiment.

Approved for Public Release; Distribution Unlimited.



(a) Path made by LKH-D (Target)

(b) Path made by LKH-D (Experiment)

Figure 38: Satellite view from UB stadium and the planned coverage paths. Virtual obstacles are shaded with diagonal lines. (a) Path planned by LKH-D algorithm. (b) Actual flight path in experiment.

Approved for Public Release; Distribution Unlimited.

justified by experimental energy measurements on an actual drone. However,this problem is NP-hard, and we decomposed it into two sub-problems. The first sub-problem divides the area among drones and has linear complexity. The second sub-problem then determines the path for each drone to cover its assigned area. Although the second sub-problem is similar to the TSP (which is NP-hard), we believe that this is acceptable as long as there are enough drones to divide the areas into reasonably sized regions (with under 50 cells) for the solver to process in a timely manner. More complex grids can be solved using sub-optimal heuristic algorithms. To this end, we adapted a heuristic for the TSP problem (LKH) and proposed the LKH-D algorithm incorporating energy consumption into the solution. We showed in simulation that our proposed heuristic is computationally faster than previously proposed algorithms and comes up with more energy-efficient paths. This section demonstrates that very sophisticated missions can be implemented on UB-ANC Drones and accurately simulated in the UB-ANC Emulator. UB-ANC Planner is available as open-source at

https://github.com/jmodares/UB-ANC-Planner.

Approved for Public Release; Distribution Unlimited.

8. *Conclusion*

As swarms of low cost attritable drones are integrated into C2ISR operations, C2 signaling and sensed data will grow exponentially, placing unprecedented demands on airborne networks. In this context, priority- and deadline-aware networking solutions will be essential to ensure that the right information is delivered to the right destination at the right time. In parallel, these same networking solutions will enable breakthroughs in a myriad of applications ranging from multimedia streaming, virtual and augmented reality, and online gaming to civilian drone networks and the internet of things.

As part of this project, we have made novel contributions related to priority- and deadline-aware scheduling, delay-sensitive medium access control, and airborne network modeling, simulation, emulation, and experimentation. For a period of 43 months, we published a total of eight conference papers [5–11], have three journal publications in preparation, and potentially one patent application.

This project has partially supported three graduate students, with one recently completing his Ph.D. dissertation. The developed work has been presented at international conferences by the PI and his students, which has enabled them to improve their presentation and communication skills, build their professional networks, and gain broader exposure to the wireless communications, networking, and robotics communities.

This project has also allowed the PI to mentor nearly 20 undergraduate students – including several from underrepresented minority groups – who contributed significantly to the experimental component of this project. This has allowed the PI to strengthen undergraduate students' technical skills and expose them to the rigors of research. These experiences helped students obtain internships and employment at AFRL in Rome, NY, launch successful careers in industry, and/or pursue graduate study.

Approved for Public Release; Distribution Unlimited.

9. *List of Acronyms*

ACU	Agent Control Unit
ANC	Airborne Networking and Communications
AODV	Ad hoc On Demand Distance Vector Routing
API	Application Programming Interface
APM	AdruPilot Mega
AWACS	Airborne Warning and Control System
BER	Bit Error Rate
C2	Command and Control
C2ISR	Command and Control, Intelligence, Surveillance, and Reconnaissance
CMDP	Constrained Markov Decision Process
CPP	Coverage Path Planning
CSMA/CA	Carrier-sense Multiple Access with Collision Avoidance
CTS	Clear to Send
CW	Congestion Window
DCF	Distributed Coordination Function
DIFS	DCF Interframe Space
DLS	Depth-limited Search
DSSS	Direct Sequence Spread Spectrum
EDF	Earliest Deadline First
EECPP	Energy-efficient Coverage Path Planning
IPC	Interprocess Communication
JSTARS	Joint Surveillance Target Attack Radar System
LB	Load Balancing
LKH	Lin-Kernighan Heuristic
LKH-D	LKH for Drones
LU	Logging Unit
MAC	Medium Access Control
MAV	Micro Aerial Vehicle
MCU	MAVLink Control Unit
MDP	Markov Decision Process
MEPP	Minimum Energy Path Planning
MIQCP	Mixed Integer Quadratic Constrained Program
MSE	Mean Squared Error
NCU	Network Control Unit

Approved for Public Release; Distribution Unlimited.

NP	Nondeterministic Polynomial Time
NS	Network Simulator
OLSR	Optimized Link State Routing Protocol
PDS	Post-decision State
PLR	Packet Loss Rate
PQ	Priority Queuing
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
RSS	Received Signal Strength
RTS	Request to Send
SCU	Sensor Control Unit
SDR	Software Defined Radio
SIFS	Short Interframe Space
SITL	Software in the Loop
SNR	Signal-to-Noise Ratio
TDMA	Time Division Multiple Access
TSP	Traveling Salesman Problem
UAS	Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle
UB	University at Buffalo
UB-ANC	University at Buffalo's Airborne Networking and Communications
USRP	Universal Software Radio Peripheral
VRP	Vehical Routing Problem
WFQ	Weighted Fair Queuing

Approved for Public Release; Distribution Unlimited.

References

[1] A. Tiwari, A. Ganguli, A. Sampath, D. S. Anderson, B.-H. Shen, N. Krishnamurthi, J. Yadegar, M. Gerla, and D. Krzysiak, "Mobility aware routing for the airborne network backbone," in *Military Communications Conference, 2008. MILCOM 2008. IEEE*. IEEE, 2008, pp. 1–7.

[2] K. D. Atherton, "Air force wants cheap attack drones it can lose in war," *Popular Science*, 2015. [Online]. Available: https://www.popsci.com/air-force-wants-cheap-attack-drones-it-can-lose-war

[3] D. Hambling, "Drone swarms will change the face of modern warfare," *Wired*, vol. 6, 2016. [Online]. Available: http://www.wired.co.uk/article/drone-swarms-change-warfare

[4] K. Namuduri, Y. Wan, M. Gomathisankaran, and R. Pendse, "Airborne network: a cyber-physical system perspective," in *Proceedings of the first ACM MobiHoc workshop on Airborne Networks and Communications*. ACM, 2012, pp. 55–60.

[5] V. Patel, N. Mastronarde, M. Medley, and J. D. Matyjas, "Towards optimal priority and deadline driven scheduling in dynamic wireless environments," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*. IEEE, 2015, pp. 1–10.

[6] N. Mastronarde, J. Modares, C. Wu, and J. Chakareski, "Reinforcement learning for energy-efficient delay-sensitive csma/ca scheduling," in *Global Communications Conference (GLOBECOM), 2016 IEEE*. IEEE, 2016, pp. 1–7.

[7] J. Modares and N. Mastronarde, "Ub-anc: a flexible airborne networking and communications testbed: poster," in *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*. ACM, 2016, pp. 95–96.

[8] J. Modares, N. Mastronarde, and K. Dantu, "Ub-anc emulator: an emulation framework for multi-agent drone networks: demo," in *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*. ACM, 2016, pp. 93–94.

[9] ——, "Ub-anc emulator: An emulation framework for multi-agent drone networks," in *Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), IEEE International Conference on*. IEEE, 2016, pp. 252–258.

Approved for Public Release; Distribution Unlimited.

[10] ——, "Realistic network simulation in the ub-anc aerial vehicle network emulator," in *IEEE INFOCOM Workshop on Wireless Communications and Networking in Extreme Environments*. IEEE, 2017, pp. 1–6.

[11] J. Modares, F. Ghanei, N. Mastronarde, and K. Dantu, "Ub-anc planner: Energy efficient coverage path planning with multiple drones," in *to appear in Proceedings of the International Conference of Robotics and Automation (ICRA '17)*, May 2017.

[12] E. Maani, P. V. Pahalawatta, R. Berry, T. N. Pappas, and A. K. Katsaggelos, "Resource allocation for downlink multiuser video transmission over wireless lossy networks," *IEEE Transactions on Image Processing*, vol. 17, no. 9, pp. 1663–1671, 2008.

[13] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, 2007.

[14] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 390–404, 2006.

[15] E. Uysal-Biyikoglu, B. Prabhakar, and A. El Gamal, "Energy-efficient packet transmission over a wireless link," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 487–499, 2002.

[16] M. A. Zafer and E. Modiano, "A calculus approach to minimum energy transmission policies with quality of service guarantees," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 1. IEEE, 2005, pp. 548–559.

[17] W. Chen, M. J. Neely, and U. Mitra, "Energy-efficient transmissions with individual packet delay constraints," *IEEE Transactions on Information Theory*, vol. 54, no. 5, pp. 2090–2109, 2008.

[18] D. Rajan, A. Sabharwal, and B. Aazhang, "Delay-bounded packet scheduling of bursty traffic over wireless channels," *IEEE Transactions on Information Theory*, vol. 50, no. 1, pp. 125–144, 2004.

[19] R. A. Berry and R. G. Gallager, "Communication over fading channels with delay constraints," *IEEE Transactions on Information Theory*, vol. 48, no. 5, pp. 1135–1149, 2002.

[20] L. Georgiadis, M. J. Neely, L. Tassiulas *et al.*, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends® in Networking*, vol. 1, no. 1, pp. 1–144, 2006.

[21] N. Mastronarde and M. van der Schaar, "Online reinforcement learning for dynamic multimedia systems," *IEEE Transactions on Image Processing*, vol. 19, no. 2, pp. 290–305, 2010.

Approved for Public Release; Distribution Unlimited.

[22] D. V. Djonin and V. Krishnamurthy, "Q-learning algorithms for constrained markov decision processes with randomized monotone policies: Application to mimo transmission control," *IEEE Transactions on Signal Processing*, vol. 55, no. 5, pp. 2170–2181, 2007.

[23] N. Salodkar, A. Bhorkar, A. Karandikar, and V. S. Borkar, "An on-line learning algorithm for energy efficient delay constrained scheduling over a fading channel," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, 2008.

[24] M. Agarwal, V. S. Borkar, and A. Karandikar, "Structural properties of optimal transmission policies over a randomly varying channel," *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1476–1491, 2008.

[25] N. Mastronarde and M. van der Schaar, "Joint physical-layer and system-level power management for delay-sensitive wireless communications," *IEEE Transactions on Mobile Computing*, vol. 12, no. 4, pp. 694–709, 2013.

[26] Y. Andreopoulos, N. Mastronarde, and M. Van Der Schaar, "Cross-layer optimized video streaming over wireless multihop mesh networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 2104–2115, 2006.

[27] N. Mastronarde, F. Verde, D. Darsena, A. Scaglione, and M. van der Schaar, "Transmitting important bits and sailing high radio waves: A decentralized cross-layer approach to cooperative video transmission," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 9, pp. 1597–1604, 2012.

[28] F. Fu and M. van der Schaar, "Structural solutions for dynamic scheduling in wireless multimedia transmission," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 5, pp. 727–739, 2012.

[29] A. Dua, C. W. Chan, N. Bambos, and J. Apostolopoulos, "Channel, deadline, and distortion (cd 2) aware scheduling for video streams over wireless," *IEEE Transactions on Wireless Communications*, vol. 9, no. 3, 2010.

[30] L. Ding, T. Melodia, S. N. Batalama, J. D. Matyjas, and M. J. Medley, "Cross-layer routing and dynamic spectrum allocation in cognitive radio ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 4, pp. 1969–1979, 2010.

[31] N. H. Mastronarde and M. van der Schaar, "A queuing-theoretic approach to task scheduling and processor selection for video-decoding applications," *IEEE Transactions on multimedia*, vol. 9, no. 7, pp. 1493–1507, 2007.

[32] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[33] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in uav communication networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1123–1152, 2016.

Approved for Public Release; Distribution Unlimited.

[34] F. Fu and M. Van Der Schaar, "A systematic framework for dynamically optimizing multi-user wireless video transmission," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 3, 2010.

[35] Y. Xiao and M. van der Schaar, "Optimal foresighted multi-user wireless video," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 1, pp. 89–101, 2015.

[36] N. Salodkar, A. Karandikar, and V. S. Borkar, "A stable online algorithm for energy-efficient multiuser scheduling," *IEEE Transactions on Mobile Computing*, vol. 9, no. 10, pp. 1391–1406, 2010.

[37] G. Bianchi, "Performance analysis of the ieee 802.11 distributed coordination function," *IEEE Journal on selected areas in communications*, vol. 18, no. 3, pp. 535–547, 2000.

[38] Y. Xiao, "Backoff-based priority schemes for ieee 802.11," in *Communications, 2003. ICC'03. IEEE International Conference on*, vol. 3. IEEE, 2003, pp. 1568–1572.

[39] K. Kim, S. Shin, and K. Kim, "A novel mac scheme for prioritized services in ieee 802.11 a wireless lan," in *ATM (ICATM 2001) and High Speed Intelligent Internet Symposium, 2001. Joint 4th IEEE International Conference on.* IEEE, 2001, pp. 196–199.

[40] J. Deng and R.-S. Chang, "A priority scheme for ieee 802. 11 dcf access method," *IEICE transactions on communications*, vol. 82, no. 1, pp. 96–102, 1999.

[41] T. H. Luan, X. Ling, and X. Shen, "Mac in motion: Impact of mobility on the mac of drive-thru internet," *IEEE Transactions on Mobile Computing*, vol. 11, no. 2, pp. 305–319, 2012.

[42] Q. Zhang and S. A. Kassam, "Finite-state markov model for rayleigh fading channels," *IEEE Transactions on communications*, vol. 47, no. 11, pp. 1688–1692, 1999.

[43] J. G. Proakis, "Digital communications." *McGraw-Hill, New York*, 1995.

[44] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data networks.* Prentice-hall Englewood Cliffs, NJ, 1987, vol. 2.

[45] L. Benini, A. Bogliolo, G. A. Paleologo, and G. De Micheli, "Policy optimization for dynamic power management," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 6, pp. 813–833, 1999.

[46] E. Altman, *Constrained Markov decision processes.* CRC Press, 1999, vol. 7.

[47] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, 2014.

[48] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11 b," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 2. IEEE, 2003, pp. 836–843.

Approved for Public Release; Distribution Unlimited.

[49] K. P. Valavanis and G. J. Vachtsevanos, *Handbook of unmanned aerial vehicles*. Springer Publishing Company, Incorporated, 2014.

[50] K. D. Haggerty, D. Wilson, G. J. Smith, T. Wall, and T. Monahan, "Surveillance and violence from afar: The politics of drones and liminal security-scapes," *Theoretical Criminology*, vol. 15, no. 3, pp. 239–254, 2011.

[51] R. L. Finn and D. Wright, "Unmanned aircraft systems: Surveillance, ethics and privacy in civil applications," *Computer Law & Security Review*, vol. 28, no. 2, pp. 184–194, 2012.

[52] C. Luo, A. P. Espinosa, D. Pranantha, and A. De Gloria, "Multi-robot search and rescue team," in *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 296–301.

[53] V. Bertram, "Unmanned surface vehicles-a survey," *Skibsteknisk Selskab, Copenhagen, Denmark*, vol. 1, pp. 1–14, 2008.

[54] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, pp. 460–466, 2015.

[55] A. Bozkurt, D. L. Roberts, B. L. Sherman, R. Brugarolas, S. Mealin, J. Majikes, P. Yang, and R. Loftin, "Toward cyber-enhanced working dogs for search and rescue," *IEEE Intelligent Systems*, vol. 29, no. 6, pp. 32–39, 2014.

[56] A. Purohit, Z. Sun, F. Mokaya, and P. Zhang, "Sensorfly: Controlled-mobile sensing platform for indoor emergency response applications," in *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*. IEEE, 2011, pp. 223–234.

[57] C. C. Murray and A. G. Chu, "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 86–109, 2015.

[58] C. C. Murray and W. Park, "Incorporating human factor considerations in unmanned aerial vehicle routing," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 4, pp. 860–874, 2013.

[59] N. Mathew, S. L. Smith, and S. L. Waslander, "Planning paths for package delivery in heterogeneous multirobot teams," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 4, pp. 1298–1308, 2015.

[60] R. D'Andrea, "Guest editorial can drones deliver?" *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 3, pp. 647–648, 2014.

[61] Z. Ball, P. Odonkor, and S. Chowdhury, "A swarm-intelligence approach to oil spill mapping using unmanned aerial vehicles," in *AIAA Information Systems-AIAA Infotech@ Aerospace*, 2017, p. 1157.

Approved for Public Release; Distribution Unlimited.

[62] M. Quaritsch, K. Kruggl, D. Wischounig-Strucl, S. Bhattacharya, M. Shah, and B. Rinner, "Networked uavs as aerial sensor network for disaster management applications," *e & i Elektrotechnik und Informationstechnik*, vol. 127, no. 3, pp. 56–63, 2010.

[63] M. Rossi, D. Brunelli, A. Adami, L. Lorenzelli, F. Menna, and F. Remondino, "Gas-drone: Portable gas sensing system on uavs for gas leakage localization," in *SENSORS, 2014 IEEE*. IEEE, 2014, pp. 1431–1434.

[64] J. Das, G. Cross, C. Qu, A. Makineni, P. Tokekar, Y. Mulgaonkar, and V. Kumar, "Devices, systems, and methods for automated monitoring enabling precision agriculture," in *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*. IEEE, 2015, pp. 462–469.

[65] J. Valente, J. Del Cerro, A. Barrientos, and D. Sanz, "Aerial coverage optimization in precision agriculture management: A musical harmony inspired approach," *Computers and electronics in agriculture*, vol. 99, pp. 153–159, 2013.

[66] L. Tang and G. Shao, "Drone remote sensing for forestry research and practices," *Journal of Forestry Research*, vol. 26, no. 4, pp. 791–797, 2015.

[67] C. Cambra, J. R. Díaz, and J. Lloret, "Deployment and performance study of an ad hoc network protocol for intelligent video sensing in precision agriculture," in *International Conference on Ad-Hoc Networks and Wireless*. Springer, 2014, pp. 165–175.

[68] Y. Pederi and H. Cheporniuk, "Unmanned aerial vehicles and new technological methods of monitoring and crop protection in precision agriculture," in *Actual Problems of Unmanned Aerial Vehicles Developments (APUAVD), 2015 IEEE International Conference*. IEEE, 2015, pp. 298–301.

[69] P. Tripicchio, M. Satler, G. Dabisias, E. Ruffaldi, and C. A. Avizzano, "Towards smart farming and sustainable agriculture with drones," in *Intelligent Environments (IE), 2015 International Conference on*. IEEE, 2015, pp. 140–143.

[70] V. Duggal, M. Sukhwani, K. Bipin, G. S. Reddy, and K. M. Krishna, "Plantation monitoring and yield estimation using autonomous quadcopter for precision agriculture," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5121–5127.

[71] B. S. Faiçal, F. G. Costa, G. Pessin, J. Ueyama, H. Freitas, A. Colombo, P. H. Fini, L. Villas, F. S. Osório, P. A. Vargas *et al.*, "The use of unmanned aerial vehicles and wireless sensor networks for spraying pesticides," *Journal of Systems Architecture*, vol. 60, no. 4, pp. 393–404, 2014.

[72] T. Le, G. Kuthethoor, C. Hansupichon, P. Sesha, J. Strohm, G. Hadynski, D. Kiwior, and D. Parker, "Reliable user datagram protocol for airborne network," in *Military Communications Conference, 2009. MILCOM 2009. IEEE*. IEEE, 2009, pp. 1–6.

Approved for Public Release; Distribution Unlimited.

[73] J. P. Rohrer, A. Jabbar, E. K. Cetinkaya, E. Perrins, and J. P. Sterbenz, "Highly-dynamic cross-layered aeronautical network architecture," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 4, pp. 2742–2765, 2011.

[74] J. Allred, A. B. Hasan, S. Panichsakul, W. Pisano, P. Gray, J. Huang, R. Han, D. Lawrence, and K. Mohseni, "Sensorflock: an airborne wireless sensor network of micro-air vehicles," in *Proceedings of the 5th international conference on Embedded networked sensor systems*. ACM, 2007, pp. 117–129.

[75] B.-N. Cheng, R. Charland, P. Christensen, A. Coyle, E. Kuczynski, S. McGarry, I. Pedan, L. Veytser, and J. Wheeler, "Characterizing routing with radio-to-router information in an airborne network," in *MILITARY COMMUNICATIONS CONFERENCE, 2011-MILCOM 2011*. IEEE, 2011, pp. 1985–1990.

[76] E. W. Frew and T. X. Brown, "Airborne communication networks for small unmanned aircraft systems," *Proceedings of the IEEE*, vol. 96, no. 12, 2008.

[77] A. Tiwari, A. Ganguli, and A. Sampath, "Towards a mission planning toolbox for the airborne network: Optimizing ground coverage under connectivity constraints," in *Aerospace Conference, 2008 IEEE*. IEEE, 2008, pp. 1–9.

[78] G. A. M. dos Santos, Z. Barnes, E. Lo, B. Ritoper, L. Nishizaki, X. Tejeda, A. Ke, H. Lin, C. Schurgers, A. Lin *et al.*, "Small unmanned aerial vehicle system for wildlife radio collar tracking," in *Mobile Ad Hoc and Sensor Systems (MASS), 2014 IEEE 11th International Conference on*. IEEE, 2014, pp. 761–766.

[79] M. Jakubiak, "Cellular network coverage analysis using uav and sdr," 2015.

[80] Y. Zhou, "Future communication model for high-speed railway based on unmanned aerial vehicles," *arXiv preprint arXiv:1411.3450*, 2014.

[81] J. Malsbury, "Modular, open-source software transceiver for phy/mac research," in *Proceedings of the second workshop on Software radio implementation forum*. ACM, 2013, pp. 31–36.

[82] A. Y. Javaid, W. Sun, and M. Alam, "Uavsim: A simulation testbed for unmanned aerial vehicle network cyber security analysis," in *Globecom Workshops (GC Wkshps), 2013 IEEE*. IEEE, 2013, pp. 1432–1436.

[83] S. Wei, L. Ge, W. Yu, G. Chen, K. Pham, E. Blasch, D. Shen, and C. Lu, "Simulation study of unmanned aerial vehicle communication networks addressing bandwidth disruptions," in *SPIE Defense+ Security*. International Society for Optics and Photonics, 2014, pp. 90 850O–90 850O.

[84] "MAVLink Protocol." [Online]. Available: http://mavlink.org

[85] "APM Planner Ground Control Station." [Online]. Available: https://github.com/diydrones/apm_planner

Approved for Public Release; Distribution Unlimited.

[86] "QGroundControl Ground Control Station." [Online]. Available: https://github.com/mavlink/qgroundcontrol

[87] "MAVLink Proxy." [Online]. Available: https://github.com/Dronecode/MAVProxy

[88] D. Hiebeler *et al.*, "The swarm simulation system and individual-based modeling," *Proceedings of Decision Support 2001: Advanced technology for natural resource management*, pp. 20–26, 1994.

[89] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan, "Mason: A multiagent simulation environment," *Simulation*, vol. 81, no. 7, pp. 517–527, 2005.

[90] B. Kate, J. Waterman, K. Dantu, and M. Welsh, "Simbeeotic: a simulator and testbed for micro-aerial vehicle swarm experiments," in *Proceedings of the 11th international conference on Information Processing in Sensor Networks*. ACM, 2012, pp. 49–60.

[91] B. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the 11th international conference on advanced robotics*, vol. 1, 2003, pp. 317–323.

[92] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, 2009, p. 5.

[93] K. Fall and K. Varadhan, "The network simulator (ns-2)," *URL: http://www. isi. edu/nsnam/ns*, 2007.

[94] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, vol. 14, 2008.

[95] X. Chang, "Network simulations with opnet," in *Simulation Conference Proceedings, 1999 Winter*, vol. 1. IEEE, 1999, pp. 307–314.

[96] X. Zeng, R. Bagrodia, and M. Gerla, "Glomosim: a library for parallel simulation of large-scale wireless networks," in *Parallel and Distributed Simulation, 1998. PADS 98. Proceedings. Twelfth Workshop on*. IEEE, 1998, pp. 154–161.

[97] "ArduPilot SITL Simulator." [Online]. Available: http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html

[98] M. Lacage and T. R. Henderson, "Yet another network simulator," in *Proceeding from the 2006 workshop on ns-2: the IP network simulator*. ACM, 2006, p. 12.

[99] G. Pei and T. Henderson, "Validation of ns-3 802.11 b phy model," *Online: http://www. nsnam. org/˜ pei/80211b. pdf*, 2009.

[100] A. Ahmadzadeh, J. Keller, G. Pappas, A. Jadbabaie, and V. Kumar, "An optimization-based approach to time-critical cooperative surveillance and coverage with uavs," in *Experimental Robotics*. Springer, 2008, pp. 491–500.

Approved for Public Release; Distribution Unlimited.

[101] I. Maza and A. Ollero, "Multiple uav cooperative searching operation using polygon area decomposition and efficient coverage algorithms," in *Distributed Autonomous Robotic Systems 6*. Springer, 2007, pp. 221–230.

[102] A. Barrientos, J. Colorado, J. d. Cerro, A. Martinez, C. Rossi, D. Sanz, and J. Valente, "Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots," *Journal of Field Robotics*, vol. 28, no. 5, pp. 667–689, 2011.

[103] C. Di Franco and G. Buttazzo, "Energy-aware coverage path planning of uavs," in *Autonomous Robot Systems and Competitions (ICARSC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 111–117.

[104] M. Torres, D. A. Pelta, J. L. Verdegay, and J. C. Torres, "Coverage path planning with unmanned aerial vehicles for 3d terrain reconstruction," *Expert Systems with Applications*, vol. 55, pp. 441–451, 2016.

[105] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[106] D. Applegate, W. Cook, S. Dash, and A. Rohe, "Solution of a min-max vehicle routing problem," *INFORMS Journal on Computing*, vol. 14, no. 2, pp. 132–143, 2002.

[107] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973.

Approved for Public Release; Distribution Unlimited.